Distinguisher for Full Final Round of Fugue-256

Jean-Philippe Aumasson and Raphael C.-W. Phan

[1] Nagravision SA, Cheseaux, Switzerland
[2] Loughborough Uni, UK

Abstract. Fugue-256 is the 256-bit version of the hash function Fugue submitted to NIST's SHA-3 competition, and selected as one of the 14 second-round candidates. Fugue-256 updates a state $S = (S_0, \ldots, S_{29})$ of 30×32=960 bits with a transform **R** that depends on a 32-bit message block and that calls once a double-AES-like round function. **R** admits trivial distinguishers, and to obtain unpredictable and pseudorandom digests, Fugue-256 only relies on a final round **G**, which maps a 960-bit state to a 256-bit digest through 18 double-AES-like rounds. The main result of this paper is an efficient distinguisher for the full 18-round **G**, building on a probability-1 differential characteristic covering 15 of those rounds. Our distinguisher finds with negligible computation pairs of inputs (S, S') that differ on 66 bits in average, and such that $\mathbf{G}(S)$ and $\mathbf{G}(S')$ remain constant for all pairs found. We also show that even if the number of rounds is increased from 18 to 30, nonrandomness remains in the final internal state of **G**. In a complete black-box setting, we furthermore show an efficient integral distinguisher for a slightly modified version of the full **G**.

Keywords: hash functions, cryptanalysis, SHA-3

1 Introduction

Among the 14 second-round candidates in NIST's SHA-3 competition [1], Fugue is the algorithm with the least third-party analysis published[3]. Submitted by Halevi, Hall and Jutla, Fugue allows formal security arguments against collision attacks and distinguishing attacks on a dedicated PRF mode. However, no formal argument is given in favor of its "random" behavior when the function is unkeyed, as in many hash function applications.

Fugue-256 (the version of Fugue with 256-bit digests) updates a state $S = (S_0, \ldots, S_{29})$ of 30×32=960 bits with a transform **R** parametrized by a 32-bit message block. **R** essentially consists of two AES-like transforms (called **SMIX**) applied to 128-bit windows of S, and thus can be easily distinguished from a random transform (for example, any difference in S_5 always propagates through **R** to S_{11}). This, plus the fact that internal collisions have been found on multiple rounds of **R** [2], indicates that **R** is weak.

To achieve notions as unpredictability and indistinguishability Fugue-256 relies instead on a much stronger transform, called the *final round* **G**, computed

[3] See the SHA-3 Zoo wiki: `http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`.

after message blocks have been processed through the **R** transform. **G** returns a 256-bit digest from the 960-bit state by making 18 double-AES-like rounds (this versus just one double-AES-like round for **R**), and unlike **R** does not admit trivial distinguishers. [3, §12.4.2] notes: "(...) one strategy that may be beneficial for the adversary is to have nonzero differential *only* in the last few columns of the state (...) after the first one or two **SMIX** steps of the final round **G** (i.e. the top right most **SMIX** steps). If this can be accomplished, then there are no active bytes in the next eight **SMIX** steps of the second loop of **G**. However, obtaining such a differential state after the first one or two **SMIX** steps of **G** is an extremely low propability event". This suggests the existence of high-probability (truncated) differential characteristics for **G**. To date, however, no distinguisher is known to exist for the full **G**.

This paper investigates distinguishers for the final round **G** of Fugue-256. Indeed, since **R** is weak, the crux of Fugue's security lies in **G**. First of all, §3 reports a distinguisher for **G**'s $G1$ rounds; then, §4 presents a *black-box* integral distinguisher for a slightly modified version of **G**, based on a detailed analysis of **G**'s propagation of multiset properties. Finally, §5 presents a differential characteristic for 15 of its 18 double-AES-like rounds, which we exploit (in §6) to construct an efficient distinguisher for the full **G**. We conclude in §7 with a discussion of our results.

Note that as **G**'s algorithm is unkeyed, this sets the stage for considering distinguishers that exhibit tuples of inputs and outputs satisfying some "evasive" property. Paraphrasing [4], such a property is easy to check but impossible to achieve with the same complexity and a non-negligible probability using oracle accesses to an ideal primitive. Such distinguishers are for example relevant to disprove indifferentiability of permutation constructions [5, 6], or to invalidate indifferentiability claims of hash constructions [7]. These distinguishers often use an "inside-out" strategy to determine inputs and outputs satisfying an evasive property; the distinguisher in §6 is inspired by that strategy.

2 Brief description of Fugue-256

Given a message m, Fugue-256 appends zeros and an 8-byte encoding of its bit length to obtain a chain of 32-bit blocks $m_0, m_1, \ldots, m_{N-1}$, which is processed by updating the state by doing $S \leftarrow \mathbf{R}(S, m_i)$ for $i = 0, \ldots, N-1$, where S is initialized to a fixed IV. The digest returned is $\mathbf{G}(S)$.

2.1 The round transformation R

Given a message word m_i, **R** transforms the 30-word state $(S_0, \ldots, S_{29})$ by doing

TIX(m_i)
ROR3; **CMIX**; **SMIX**
ROR3; **CMIX**; **SMIX**

where **ROR3** right-rotates S by 3 words (i.e., it simultaneously sets $S_i \leftarrow S_{i-3}$ for all $i = 0, \ldots, 29$), and where $\mathbf{TIX}(m_i)$ does

$$S_{10} += S_0;\ S_0 = m_i;\ S_8 += S_0;\ S_1 += S_{24}$$

and where **CMIX** does

$$S_0 += S_4;\ S_1 += S_5;\ S_2 += S_6;$$
$$S_{15} += S_4;\ S_{16} += S_5;\ S_{17} += S_6$$

The **SMIX** function is described in §2.3.

Note that here additions are in fact bitwise XORs. A message word affects 11 state words through **R**, namely $S_0 \ldots, S_6$ and $S_{14}, \ldots, S_{17}$. In particular, S_{14} equals the sum of the initial S_8 and of m_i.

Also, note that two states that differ only by S_{10} and S_0 such that $S_{10} + S_0$ is unchanged map to the same value, regardless of the block m_i. A detailed analysis in [3, §10] shows that this property is necessary to obtain an internal collision after four applications of **R**, along with the conditions that

- $S_1 + S_{24}$ is unchanged,
- $S_3, S_7, S_{12}, S_{15}, S_{16}$, and S_{17} have nonzero difference,
- S_5, S_6, and S_{11} have zero difference.

[3, §10] also demonstrates that finding internal collisions is difficult.

2.2 The final round G

The final round **G** transforms the internal state $(S_0, \ldots, S_{29})$ by doing five $G1$ *rounds*:

ROR3; **CMIX**; **SMIX**
ROR3; **CMIX**; **SMIX**

followed by 13 $G2$ *rounds*:

$$S_4 += S_0; S_{15} += S_0;\ \mathbf{ROR15};\ \mathbf{SMIX}$$
$$S_4 += S_0; S_{16} += S_0;\ \mathbf{ROR14};\ \mathbf{SMIX}$$

where **ROR15** and **ROR14** right-rotate S by 15 and 14 words, respectively.

G finally returns as hash value the eight words

$$S_1, S_2, S_3, (S_4 + S_0), (S_{15} + S_0), S_{16}, S_{17}, S_{18}$$

G thus makes in total 36 calls to **SMIX**.

We shall henceforth refer to **G**'s two types of rounds as "$G1$ rounds" and "$G2$ rounds". S_i^j will denote the value of S_i at the input of **G**'s round $(j + 1) \geq 1$; for example, S_1^0 is the second word of the initial state.

2.3 The permutation SMIX

SMIX transforms the 128-bit vector (S_0, S_1, S_2, S_3) and *lets all the other S_i's unchanged*. Inspired by the AES round function, **SMIX** views its input as a 4×4 matrix of bytes. First each byte passes through the AES S-box, then the Super-Mix linear transformation is applied. Unlike AES' `MixColumn`, Super-Mix operates on the whole 128-bit state rather than on each column independently, which makes **SMIX** arguably stronger than the original AES round. Note that Super-Mix is the only Fugue component that provides bytewise mixing within word borders, the other Fugue components only provide wordwise mixing.

We refer to [3] for a detailed description of Super-Mix, and in particular of its matrix representation **N**.

2.4 Previous analysis of Fugue-256

To date, only two references have reported analyses of Fugue: the submission document by Fugue designers [3] and the third-party analysis by Khovratovich [2]. We briefly summarize their main results:

Diffusion of R. Diffusion properties for **R** were analyzed by considering adversaries injecting differences into the input message block m_i [3, §8.1], or into the input state S [3, §8.2] of **R**. The latter analysis concluded that after five consecutive applications of **R**, the output bytes from **SMIX** have not been non-linearly influenced by all 120 input state bytes of S. This illustrates the weak diffusion properties of **R**, and is one reason why the final round **G** bears paramount importance.

Diffusion of G. [3, §8.3] claims that after the first part of **G** (i.e., the five $G1$ rounds), "Every 'final byte' in columns 0-4 (and 15) depends non-linearly on all the 120 'initial bytes' ...". Further, [3, §8.4] observes that after the full **G** all output bytes depend nonlinearly on all 120 input bytes via more than 25 **SMIX**-es.

Internal collisions. [3, §10] analyses in detail Fugue-256's resistance to internal collisions, with as main results the necessary conditions for an internal collision (Lemma 10.1) and the upper bound 2^{-246} on the probabilty of an internal collision after four message injections, given a random initial state and any fixed non-zero difference in it (Theorem 10.2). Note that the best known internal collision search method [2] for Fugue-256 runs in 2^{352}.

External collisions. [3, §11] studies external collisions for Fugue-256 (i.e., collisions occuring only after the final round **G**), and gives the bound 2^{-129} on the probability of an external collision for the second part of **G** (i.e., the 13 $G2$ rounds).

PRF analysis of Fugue. [3, §12.4] considers the security of a proposed PRF construction based on Fugue-256, which sets a 256-bit key as the initial $(S_{22}, \ldots, S_{29})$. The analysis was performed notably via differential cryptanalysis and against a collision-finding adversary, assuming that the best adversarial approach is to look for partial collisions. It was further claimed that it is difficult to launch distinguishing attacks with probability greater than 2^{-128}. They considered the adversary inducing nonzero differences into the last few words (columns) of the state after up to the first round of **G**, and then remarked that this should propagate through the five $G1$ rounds and most of $G2$ rounds.

3 Partitioning distinguisher for $G1$ rounds

We present a distinguisher that applies to the five $G1$ rounds of **G** (also called "TIX-less rounds" in [3]). These $G1$ rounds offer more diffusion than $G2$ rounds, i.e., **CMIX** in a $G1$ round involving three XORs diffuses more words than the two XORs in a $G2$ round. We start by tracing the propagation of the input word S_5^0 through **G**, where we denote the bytes of S_5^0 as $\mathtt{B}_0\mathtt{B}_1\mathtt{B}_2\mathtt{B}_3$. This S_5^0 propagates with probability one to S_{29}^4. In round 5, **ROR3** moves this to position S_2, which then enters **SMIX**. Let the bytes after the S-box be $b_0b_1b_2b_3$. The definition of the matrix **N** of Super-Mix leads to output words $S_0S_1S_2S_3$ of the following form:

- S_0: $f(b_0)$ $f(b_1)$ $f(b_0b_1b_2b_3)$ $f(b_3)$
- S_1: $f(b_0)$ $f(b_0b_2b_3)$ 0 $f(b_3)$
- S_2: $f(b_0b_1b_2b_3)$ $f(b_1)$ 0 $f(b_3)$
- S_3: $f(b_0)$ $f(b_1)$ 0 $f(b_0b_1b_2b_3)$

where $f(\cdot)$ denotes some undetermined function of the input byte(s) and where "0" means no influence of any of the input b_i.

After **ROR3**, these words become $S_3S_4S_5S_6$ and then **CMIX** XORs $S_4S_5S_6$ with $S_0S_1S_2$ (which up to this point had not been influenced by S_5^0 at all), to form the new $S_0S_1S_2$. Thus, $S_0S_1S_2S_3$ entering the second **SMIX** of round 5 are influenced by the the same bytes of S_5^0 that had entered the first **SMIX** of round 5, except the order is shifted, i.e. the $S_0S_1S_2S_3$ inputs to **SMIX** at this point are of the form:

- S_0: $f(b_0)$ $f(b_0b_2b_3)$ 0 $f(b_3)$
- S_1: $f(b_0b_1b_2b_3)$ $f(b_1)$ 0 $f(b_3)$
- S_2: $f(b_0)$ $f(b_1)$ 0 $f(b_0b_1b_2b_3)$
- S_3: $f(b_0)$ $f(b_1)$ $f(b_0b_1b_2b_3)$ $f(b_3)$

and after **SMIX** we have $S_0S_1S_2S_3$ of the form:

- S_0: $f(b_0b_1b_2b_3)$ $f(b_0b_1b_2b_3)$ $f(b_0b_1b_2b_3)$ $f(b_0b_1b_2b_3)$,
- S_1: $f(b_0b_1b_3)$ $f(b_0b_1b_2b_3)$ $f(b_0b_1b_3)$ $f(b_0b_1b_2b_3)$,
- S_2: $f(b_0b_1b_2b_3)$ $f(b_0b_1b_2b_3)$ $f(b_0b_1b_2b_3)$ $f(b_0b_1b_2b_3)$,
- S_3: $f(b_0b_1b_2b_3)$ $f(b_0b_1b_2b_3)$ $f(b_0b_1b_2b_3)$ $f(b_0b_1b_2b_3)$,

i.e. the first and third bytes of the **SMIX** output word at S_1^5 are only influenced by $b_0 b_1 b_3$ and *not* by b_2.

Our distinguisher consists in obtaining a pair of inputs to **G** such that only the byte $\mathtt{B}_2$ of S_5^0 varies while the other bytes are constant, and checking that after five rounds the first and third bytes in S_1^5 remain unchanged. Note that this does not require the cryptanalyst to know what the input values to **G** are, except that the byte $\mathtt{B}_2$ is variable. This is akin to a partition on the input space (where only the byte $\mathtt{B}_2$ of S_5^0 is non-constant) that leads to an output partition of colliding values (first and third bytes in S_1^5 are constant) indexed by these two constant byte positions. Such a colliding partition bears some resemblance to the attack by Knudsen on SAFER K-64 [8].

This distinguisher succeeds with probability one and is sufficient to distinguish the five $G1$ rounds from random.

This result is surprising given that the same bytes of S_5^0 enter **SMIX** twice within round 5 and yet do not fully influence all bytes of the final **SMIX** output.

4 Integral distinguishers for G variants

This section presents integral distinguishers for variants of the **G** function. Integral distinguishers were first introduced by Knudsen on the Square cipher [9], and work by tracing a multiset of words through round components and exploiting the fact that certain operations (e.g., bijections), preserve the zero integral sum of the multiset. A multiset can be defined to have the following properties:

- Permutation (P): a w-bit P multiset has 2^w unique elements $p_i \in \{0,1\}^w$.
- Constant (C): a w-bit C multiset has 2^w constant elements $c_i \in \{0,1\}^w$.

Note that the sum of either multiset is zero, i.e., $\bigoplus_{i=0}^{2^w-1} p_i = \bigoplus_{i=0}^{2^w-1} c_i = 0$.

In addition, we denote by B (balanced) a multiset whose 2^w elements sum to zero, but that is not P nor C. Also, we denote thereafter $S_0 \sim S_3$ for $S_0 S_1 S_2 S_3$.

4.1 Integrals through SMIX

G uses components that preserve byte boundaries, hence it is natural to consider bytewise integral distinguishers. The main component to analyze is **SMIX**. In fact, it is sufficient to concentrate on Super-Mix since the S-box preserves both P and C. Recall that Super-Mix just involves multiplication of input bytes with a constant (1, 4, 5, 6 or 7) and XORs of bytes.

Consider a multiset of 256 values $S_0 S_1 S_2 S_3$ on input to Super-Mix, such that S_2 has at least one P byte and the remaining (if any) are C bytes, while S_0, S_1, and S_3 have only C bytes. We denote the byte multisets of S_2 as $\mathtt{B}_0\mathtt{B}_1\mathtt{B}_2\mathtt{B}_3$, $\mathtt{B}_i \in \{\mathtt{P}, \mathtt{C}\}$. Then, the output multiset of Super-Mix, i.e., the new $S_0 S_1 S_2 S_3$, is of the form:

$\mathtt{B}_0 + 0 + 0 + 0 + \mathtt{C}$	$\mathtt{B}_0 + 0 + 0 + 0 + \mathtt{C}$	$6\mathtt{B}_0 + 4\mathtt{B}_1 + 7\mathtt{B}_2 + \mathtt{B}_3 + \mathtt{C}$	$4\mathtt{B}_0 + 0 + 0 + 0 + \mathtt{C}$
$0 + \mathtt{B}_1 + 0 + 0 + \mathtt{C}$	$\mathtt{B}_0 + 0 + 4\mathtt{B}_2 + 7\mathtt{B}_3 + \mathtt{C}$	$0 + 7\mathtt{B}_1 + 0 + 0 + \mathtt{C}$	$0 + 4\mathtt{B}_1 + 0 + 0 + \mathtt{C}$
$7\mathtt{B}_0 + \mathtt{B}_1 + \mathtt{B}_2 + 4\mathtt{B}_3 + \mathtt{C}$	$0 + \mathtt{C}$	$0 + \mathtt{C}$	$0 + \mathtt{C}$
$0 + 0 + 0 + \mathtt{B}_3 + \mathtt{C}$	$0 + 0 + 0 + \mathtt{B}_3 + \mathtt{C}$	$0 + 0 + 0 + 7\mathtt{B}_3 + \mathtt{C}$	$4\mathtt{B}_0 + 7\mathtt{B}_1 + \mathtt{B}_2 + 5\mathtt{B}_3 + \mathtt{C}$

If the input S_2 has all P bytes, i.e., $\mathtt{B}_i = \mathtt{P}$ for $i = 0 \ldots 3$, then we have the output byte multisets of $S_0 \sim S_3$ with the properties

$$\mathtt{PPBP} \quad \mathtt{PBCP} \quad \mathtt{BPCP} \quad \mathtt{PPCB}\ .$$

Since **G** respects byte boundaries, we can do better by starting with only one P multiset. More precisely, let the input multiset be such that S_2 has only one P byte and the other bytes are C, while the other S_i only have C bytes. The output multiset after **SMIX** will be as follows, depending on the location of the P byte.

- $S_2 \equiv \mathtt{PCCC} \Rightarrow S_0S_1S_2S_3 \equiv \mathtt{PCPC\ PPCC\ PCCC\ PCCP}$.
- $S_2 \equiv \mathtt{CPCC} \Rightarrow S_0S_1S_2S_3 \equiv \mathtt{CPPC\ CCCC\ PPCC\ CPCP}$.
- $S_2 \equiv \mathtt{CCPC} \Rightarrow S_0S_1S_2S_3 \equiv \mathtt{CCPC\ CPCC\ PCCC\ CCCP}$.
- $S_2 \equiv \mathtt{CCCP} \Rightarrow S_0S_1S_2S_3 \equiv \mathtt{CCPP\ CPCP\ PCCP\ CCCP}$.

4.2 Integrals through 5.5 G rounds

We now trace multiset properties through the first 5.5 rounds of **G**, i.e., five $G1$ rounds and half a $G2$ round. Without loss of generality, we take as example $S_2 = \mathtt{PCCC}$ input to **SMIX**. The analysis similarly applies for P in other byte positions of S_2 input to **SMIX**. We start with a multiset of 256 values at the input to **G** such that they have C bytes in all byte positions except in the first byte of S_5, which is a P byte. Then we know (from our analysis in §3) that this propagates undiffused through to the output of round 4, where the P byte is now in the first byte position of S_{29}. Going into round 5, through **ROR3**, S_{29} becomes S_2. **CMIX** affects S_2 via S_6 but S_2's multiset properties are preserved because S_6 is all C bytes, thus S_2 remains as PCCC. Going through **SMIX**, the output $S_0S_1S_2S_3$ becomes

$$\mathtt{PCPC} \quad \mathtt{PPCC} \quad \mathtt{PCCC} \quad \mathtt{PCCP}\ .$$

All other S_i's have only C bytes. Going through **ROR3**, $S_0S_1S_2S_3$'s PCPC PPCC PCCC PCCP goes into $S_3S_4S_5S_6$. Going through **CMIX**, $S_0S_1S_2$ and $S_{15}S_{16}S_{17}$ are influenced by $S_4S_5S_6$ (though byte boundaries are still respected) so $S_0S_1S_2$ and $S_{15}S_{16}S_{17}$ also become PPCC PCCC PCCP. Going through **SMIX**, recall that $S_0S_1S_2S_3$ on input to this is PPCC PCCC PCCP PCPC. The S-box preserves the byte multiset properties, thus Super-Mix is the critical operation; the output

from this **SMIX** will have the byte multiset properties as:

$$
\begin{aligned}
&(\mathtt{P} + 4\mathtt{P} + 7\mathtt{C} + \mathtt{C}) + \mathtt{P} + \mathtt{P} + \mathtt{P} = \mathtt{B} \\
&\mathtt{P} + (\mathtt{P} + \mathtt{C} + 4\mathtt{C} + 7\mathtt{C}) + \mathtt{C} + \mathtt{C} = \mathtt{B} \\
&\mathtt{C} + \mathtt{C} + (7\mathtt{P} + \mathtt{C} + \mathtt{C} + 4\mathtt{P}) + \mathtt{P} = \mathtt{B} \\
&\mathtt{C} + \mathtt{C} + \mathtt{P} + (4\mathtt{P} + 7\mathtt{C} + \mathtt{P} + \mathtt{C}) = \mathtt{B}
\end{aligned}
$$

$$
\begin{aligned}
&0 + (4\mathtt{C} + 7\mathtt{C} + \mathtt{C}) + \mathtt{P} + \mathtt{P} = \mathtt{B} \\
&\mathtt{P} + 0 + (\mathtt{P} + 4\mathtt{C} + 7\mathtt{P}) + \mathtt{C} = \mathtt{B} \\
&\mathtt{C} + \mathtt{C} + 0 + (7\mathtt{P} + \mathtt{C} + 4\mathtt{C}) = \mathtt{P} \\
&(4\mathtt{P} + 7\mathtt{P} + \mathtt{C}) + \mathtt{C} + \mathtt{P} + 0 = \mathtt{B}
\end{aligned}
$$

$$
\begin{aligned}
&0 + 7\mathtt{P} + (6\mathtt{P} + 4\mathtt{C} + 7\mathtt{C} + \mathtt{P}) + 7\mathtt{P} = \mathtt{B} \\
&7\mathtt{P} + 0 + 7\mathtt{C} + (\mathtt{P} + 6\mathtt{C} + 4\mathtt{P} + 7\mathtt{C}) = \mathtt{B} \\
&(7\mathtt{P} + \mathtt{P} + 6\mathtt{C} + 4\mathtt{C}) + \mathtt{C} + 0 + 7\mathtt{P} = \mathtt{B} \\
&7\mathtt{C} + (4\mathtt{P} + 7\mathtt{C} + \mathtt{C} + 6\mathtt{C}) + 7\mathtt{P} + 0 = \mathtt{B}
\end{aligned}
$$

$$
\begin{aligned}
&0 + 4\mathtt{P} + 4\mathtt{P} + (5\mathtt{P} + 4\mathtt{C} + 7\mathtt{P} + \mathtt{C}) = \mathtt{B} \\
&(\mathtt{P} + 5\mathtt{P} + 4\mathtt{C} + 7\mathtt{C}) + 0 + 4\mathtt{C} + 4\mathtt{C} = \mathtt{B} \\
&4\mathtt{C} + (7\mathtt{P} + \mathtt{C} + 5\mathtt{C} + 4\mathtt{C}) + 0 + 4\mathtt{P} = \mathtt{B} \\
&4\mathtt{C} + 4\mathtt{C} + (4\mathtt{P} + 7\mathtt{C} + \mathtt{C} + 5\mathtt{P}) + 0 = \mathtt{B}
\end{aligned}
$$

So, at the end of round 5, we have $S_0S_1S_2S_3$ of the form

$$\mathtt{BBBB} \quad \mathtt{BBPB} \quad \mathtt{BBBB} \quad \mathtt{BBBB}\,,$$

and both $S_4S_5S_6$ and $S_{15}S_{16}S_{17}$ are PPCC PCCC PCCP. All other S_i's have only C bytes. We now proceed into round 6:

- $S_4 += S_0; S_{15} += S_0$: only S_4 and S_{15}'s multiset properties are changed:

$$
\begin{aligned}
S_4 &= S_4 + S_0 \equiv \mathtt{PPCC} + \mathtt{BBBB} \equiv \mathtt{BBBB} \\
S_{15} &= S_{15} + S_0 \equiv \mathtt{PPCC} + \mathtt{BBBB} \equiv \mathtt{BBBB}\,.
\end{aligned}
$$

  Other S_i's are unchanged.
- **ROR15**: we trace only the S_i's that have P or B bytes, i.e. $S_0 \sim S_6 \to S_{15} \sim S_{21}$, $S_{15} \sim S_{17} \to S_0 \sim S_2$. All other S_i's are C bytes.

– **SMIX**: the input $S_0S_1S_2S_3$ is BBBB PCCC PCCP CCCC so the output is of the form (note that the S-box destroys a B property)

$$(?) + \mathtt{P} + \mathtt{P} + \mathtt{C} = ?$$
$$? + (\mathtt{P} + \mathtt{C} + 4\mathtt{C} + 7\mathtt{C}) + \mathtt{C} + \mathtt{C} = ?$$
$$? + \cdots = ?$$
$$? + \cdots = ?$$

$$0 + (4\mathtt{C} + 7\mathtt{C} + \mathtt{C}) + \mathtt{P} + \mathtt{C} = \mathtt{P}$$
$$? + \cdots = ?$$
$$? + \cdots = ?$$
$$? + \cdots = ?$$

$$0 + 7\mathtt{P} + (6\mathtt{P} + 4\mathtt{C} + 7\mathtt{C} + \mathtt{P}) + 7\mathtt{C} = \mathtt{B}$$
$$? + \cdots = ?$$
$$? + \cdots = ?$$
$$? + \cdots = ?$$

$$0 + 4\mathtt{P} + 4\mathtt{P} + (5\mathtt{C} + 4\mathtt{C} + 7\mathtt{C} + \mathtt{C}) = \mathtt{B}$$
$$? + \cdots = ?$$
$$? + \cdots = ?$$
$$? + \cdots = ?$$

Thus, the output $S_0S_1S_2S_3$ is ???? P??? B??? B???. Here, "?" denotes a byte multiset where it is not known if it has B, C, or P properties.

Note here that we concentrate only on tracing the multiset properties in the *output* $S_0S_1S_2S_3$. This is because these four words undergo the most diffusion through the rounds and because the S_0 output is preserved intact through many subsequent rounds, e.g. the S_0 output of round 6 is observable via the XOR of two output words after round 18.

4.3 Integrals through 18 rounds minus an XOR and a Super-Mix

We extend the above 5.5-round integral distinguisher to 18 rounds of a slightly modified version of **G**, wherein only round 6's second half is tweaked. More precisely, the second half of round 6 omits the $S_{16} += S_0$ operation, as well as Super-Mix in **SMIX** of this second half (but the S-box is *not* omitted), and where the final round 18 output includes S_{14}. The 5.5-round integral distinguisher of the previous subsection can continue through this half-round to the end of round 6 as follows.

– Only $S_4 += S_0$: only S_4's multiset properties are changed, i.e. S_4 = ???? since it is XORed with S_0 which is ????. Other S_i's are unchanged.

– **ROR14**: we trace only the S_i's with properties we are interested in, i.e., $S_0 \sim S_3 \rightarrow S_{14} \sim S_{17}$, $S_4 \rightarrow S_{18}$, $S_{15} \rightarrow S_{29}$, $S_{16} \rightarrow S_0$, $S_{17} \sim S_{21} \rightarrow S_1 \sim S_5$.
– **SMIX** minus Super-Mix: the input $S_0S_1S_2S_3$ is `BBPB BBBB BBBB BBBB` so the output (note that the S-box destroys a `B` property) $S_0S_1S_2S_3$ is ??`P`? ???? ???? ???? ????.

We only exploit the byte multisets of S_0 from this point onwards.

The key observation is that the output integral sum after all 18 rounds of **G** in positions S^{18}_{19+i}, $i = 0, 1, \ldots, 10$, equals the integral sum in S_0^{7+i} after the first $7 + i$ rounds of **G**. This means that the output integral sum allows to check the particular S_0 property deep within the **G** structure, as deep as the output of the first seven rounds. Going deeper, the integral sum in S_0^6 at the output of the first six rounds equals that of S^{17}_{19} (at the output of round 17). A more careful analysis of round 18 reveals that $S^{17}_{19} = S^{18}_{18} + S^{18}_{14}$. That is, by simply XORing two output integral word sums after round 18, one can compute the integral sum in S^{17}_{19} and hence the integral sum in S_0^6 at the output of round 6.

This leads to the following distinguisher for 18 rounds of this **G** variant:

1. Collect a multiset of 256 inputs to round 1 such that the first byte of the S_5 input is `P`; and all other bytes of S_i are `C`, and obtain corresponding outputs from this **G** variant oracle.
2. Compute the XOR between the integral sums of S^{18}_{18} and S^{18}_{14} and check for ??`P`?, i.e. the third byte integral should be `P`.

In Appendix, Fig. 4 shows the evolution of the multiset properties.

These integral results have been experimentally verified with the official source code of Fugue. Note that this distinguisher works in a black-box way, i.e., the adversary does not manipulate internal state values nor require them for computation of desired inputs of the distinguisher. Furthermore, the distinguisher is applicable even if the cryptanalyst does not know the inputs to **G**. One just needs to ensure that all 256 inputs are identical, except for the S_5 byte which is distinct for each input.

5 Probability-1 characteristics for 15 G rounds

In the Fugue analysis in [3, §12.4.2], a difference is considered to be induced in the right half of the input state such that after one $G1$ round, differences exist in the *last* few words. This difference propagation was then traced through 5 $G1$ rounds before $G2$ and it was concluded that if done this way then no differences enter the **SMIX** for most of the subsequent $G2$ rounds.

Given an arbitrary state S, we consider instead an arbitrary input difference Δ in S_5 (left half of the state) and no difference in the other S_i's. Below we show that after four $G1$ rounds followed by 11 $G2$ rounds, the state S always has a difference Δ in S_{18} and no difference in other S_i's. This is illustrated for $\Delta =$ `FFFFFFFF` in Fig. 1.

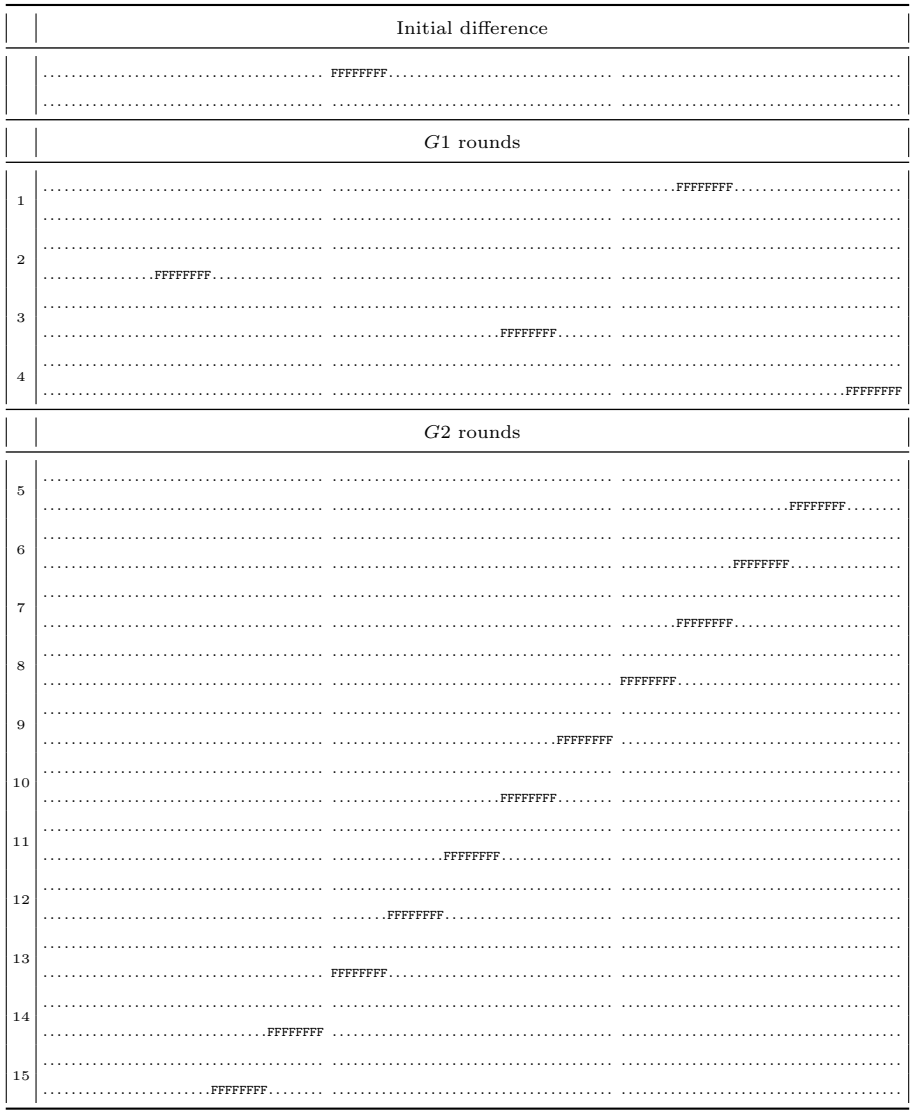

Fig. 1. Evolution of differences given an initial difference `FFFFFFFF` in S_5, with 4 $G1$ rounds and 11 $G2$ rounds.

5.1 Propagation through $G1$ rounds

During the four $G1$ rounds, a difference in S_5 does not activate any **CMIX** nor **SMIX** (i.e., the difference never enters those functions). This is because after all the evaluations **ROR3**, the difference is never moved to a word of index in $\{0, \ldots, 6\}$, which are the indices of words entering **CMIX** or **SMIX**. Indeed, going through each of the 4 rounds, **ROR3** respectively moves Δ to

1. S_8 and S_{11},
2. S_{14} and S_{17},
3. S_{20} and S_{23},
4. S_{26} and S_{29}.

Note that an additional $G1$ round would move Δ to S_2 and so would activate an **SMIX**. The probability-1 characteristic thus stops here for the $G1$ rounds.

5.2 Propagation through $G2$ rounds

As for $G1$ rounds, we show that the initial difference (now in S_{29}) never activates **SMIX** nor the additions $S_4 += S_0$, $S_{15} += S_0$, $S_{16} += S_0$. That is, Δ is never moved to a word of index in $\{0, \ldots, 3\}$. Going through each of the 11 rounds, **ROR15** and **ROR14** respectively move Δ to S_{14} and S_{28}, S_{14} and S_{28}, S_{13} and S_{27}, ..., S_5 and S_{19}, and finally to S_4 and S_{18}. Again, an additional round would activate **SMIX**, as Δ would enter S_3.

6 Distinguisher for full G

This section describes the actual distinguisher for the full 18 rounds of **G**. This distinguisher consists in an efficient method to find pairs of initial states (S, S') that differ on 66 bits in average, and such that $\mathbf{G}(S)$ and $\mathbf{G}(S')$ remain constant for all pairs found. Note that distinct pairs satisfying only the latter condition are trivial to find for **G** (by inverting the algorithm for some well-chosen final states), however the former condition would be satisfied with negligible probability.

We shall naturally exploit the differential characteristic described in §5, and follow an "inside-out" strategy, as suggested by the observation below.

6.1 Basic observation: inside-out strategy

First, observe that one can choose the value of any intermediate state between (before) the first and (after) the last round, and deduce the corresponding initial and final states by computing backwards and forwards. In particular, one can choose an intermediate state between (after) the first and (before) the 17-th round, and determine the position of a difference in S to follow the differential characteristic depicted on Fig. 1, and deduce the corresponding initial and final states.

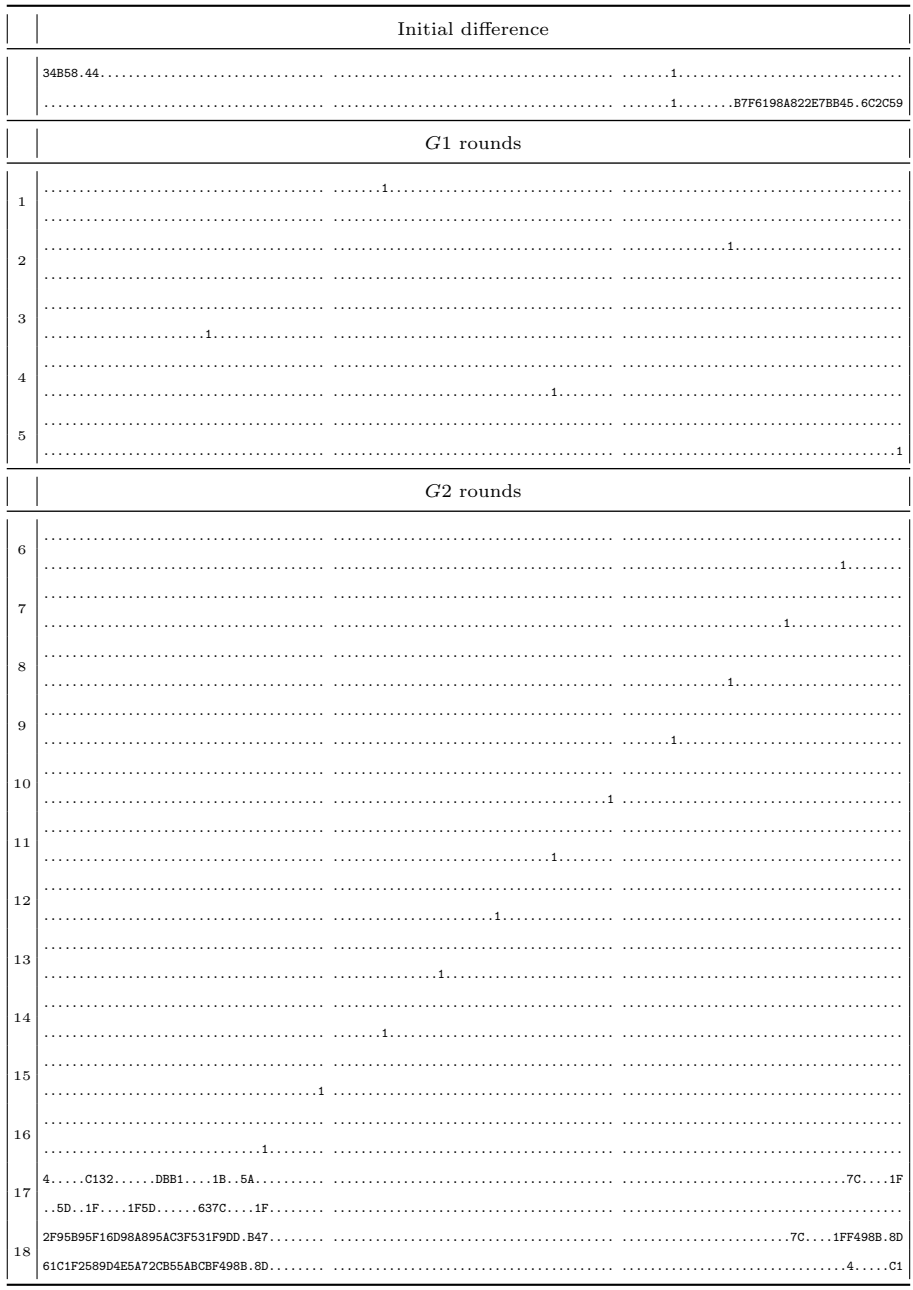

Fig. 2. Evolution of differences with a difference `00000001` in the 15 intermediate rounds, and a state S set to zero before the 17th round. A state is displayed on two lines left-to-right from S_0 to S_{14}, and from S_{15} to S_{29}, in hexadecimal basis, replacing zeroes by dots for readability.

6.2 Forwards: constant $\mathbf{G}(S)$ and $\mathbf{G}(S')$

Recall that after its last (18th) round, **G** extracts 256 bits from the 960-bit final state by returning $S_1, S_2, S_3, (S_4 + S_0), (S_{15} + S_0), S_{16}, S_{17}, S_{18}$. Because the probability-1 characteristic activates **SMIX** at the 17th round, all the words above will depend on Δ. However, they do not depend on the *values* of all the state words entering the 17th round. This observation allows us to find the value of distinct pairs of states (S, S') *before the 17th round* such that their respective digests are constant for all pairs found (and thus their difference is constant as well).

A straightforward analysis shows that the digest returned does *not* depend on the values of $S_7 \sim S_{13}$ and $S_{21} \sim, S_{28}$ entering the 17th round. It is thus sufficient to fix $S_0 \sim S_6$, $S_{14} \sim S_{20}$, and S_{29} to ensure that $\mathbf{G}(S)$ and $\mathbf{G}(S')$ remain unchanged, by varying the other words (recall that S' is S after setting a difference in S_{18}).

6.3 Backwards: sparse initial differences

Once one has determined two states entering the 17th round with a difference Δ in S_{18}, it is guaranteed that after inverting 11 $G2$ rounds and then four $G1$ rounds, one obtains two states with only a difference Δ in S_5 (as one follows that 15-round characteristic with probability 1). Since there is only one $G1$ round left to invert in order to get to the initial state, most of the state is unaffected by Δ: it is easy to show that the initial state will have difference Δ in S_{10} and S_{25}, caused by the first inverse **CMIX**, and undetermined differences in $S_0, S_{27}, S_{28}, S_{29}$, caused by the second inverse **SMIX**.

Therefore, only six words of the initial state have nonzero differences, and if Δ is chosen of minimal Hamming weight (1), then the two states have in average 66 bit differences (assuming that the inverse **SMIX** causes in average differences in half the bits of its input; this seems a reasonable assumption, as the linear layer of the inverse **SMIX** has a much denser algebraic description than the original, as shown in [3, Fig. 5]). Our experiments did not contradict that estimate.

6.4 Example

Fig. 2 shows the evolution of differences when the S entering the 17th round is set to zero (and thus S' is all zero as well, except for the LSB of S_{18}). From the above observation, varying $S_7 \sim S_{13}$ and $S_{21} \sim S_{28}$ does not affect the final differences. In particular, if $S_0 \sim S_6$ and $S_{14} \sim S_{20}$ are zero when entering the 17th round, the digests returned from S and S' are always

```
588CA4D5 2283E13E 786018F3 5B300BF9 4AC5F765 D5DD2538 C74BE4BB 62571AA9
```

and

```
4E552E5C 7840140F 81BD13B4 74A5B2A6 0491BC62 48937F4A 0C1E4F70 96CFAA24
```

regardless of the values of other S_i's.

6.5 Extensions

We briefly describe extensions of our distinguisher to "more than **G**": First, the distinguisher for **G** can be extended to a distinguisher for $\mathbf{G} \circ \mathbf{R}$, since after an inverse **R** the state obtained remains relatively sparse (as the two inverse **CMIX** are inactive).

If the number of $G2$ rounds is increased, there remain fixed differences in the output of **G** after up to 20 rounds (i.e., in $G1$ plus 15 $G2$ rounds). Nonrandomness *in the internal state* can be observed after at least 30 rounds (5 plus 25 rounds, that is, after 60 **SMIX**-es), because constant differences computed after 20 rounds follow the probability-1 characteristic 10 further rounds (see Fig. 3).

	$G2$ rounds (continued)		
18	2F95B95F16D98A895AC3F531F9DD.B47.....	.....	.....7C....1FF498B.8D
	61C1F2589D4E5A72CB55ABCBF498B.8D.....	.....	.....4.....C1
19	.C.765.A.CBC24976.7BC6FFE6A968A1.....	.....	.....7C....1FF498B.8D352B.A8D
	DC5139C5689E98EBB92F1FEC352B.A8D.....	.....	.....4.....C12F95B95F
20	38.354D8D19CAD1ADD3C21C8E8623.1F.....	.....	.....7C....1FF498B.8D352B.A8DBCAFE2.2
	8FF6CD4DA2.B8.6DB6C.D8DDBCAFE2.2.....	.....	.....4.....C12F95B95F.C.765.A
21	A6B5FF.87BE5E7.73219688B7FFC6C3A.....	.....	7C....1FF498B.8D352B.A8DBCAFE2.2CAF8A797
	7163.6E1BE669A322DA32653CAF8A797.....	.....	.....4.....C12F95B95F.C.765.A38.354D8
22	DAC69612873CEFA23.A839349B849765.....	.....7C....1F	F498B.8D352B.A8DBCAFE2.2CAF8A7972ECBE532
	AD64ED4681F9.BE7E5D181CB2ECBE532.....	.....	4.....C12F95B95F.C.765.A38.354D8A6B5FF.8
23	9F2848C91D1466FFD89A93E9AECC9C9D.....	.....7C....1FF498B.8D	352B.A8DBCAFE2.2CAF8A7972ECBE532929EA61D
	B517.1EDF8E2E84.C937.923929EA61D.....	.....4.....C1	2F95B95F.C.765.A38.354D8A6B5FF.8DAC69612
24	DF1D915133ED562FB15FFD41CB82D8F9.....	.....7C....1FF498B.8D352B.A8D	BCAFE2.2CAF8A7972ECBE532929EA61D8.83AE13
	42A417718.2A2D.84E4479DB8.83AE13.....	.....4.....C12F95B95F	.C.765.A38.354D8A6B5FF.8DAC696129F2848C9
25	5A1AB2BD39816966DD42511FDD7B8613.....	.....7C....1FF498B.8D352B.A8DBCAFE2.2	CAF8A7972ECBE532929EA61D8.83AE13.8C394B1
	44.7DF3CAB9812A38E359E12.8C394B1.....	.....4.....C12F95B95F.C.765.A	38.354D8A6B5FF.8DAC696129F2848C9DF1D9151
26	771D7EAFCE558D.1C3567BEC.F..74A8.....	7C....1FF498B.8D352B.A8DBCAFE2.2CAF8A797	2ECBE532929EA61D8.83AE13.8C394B154861.AD
	7B6CC2ED714A24192B78B97.54861.AD.....	.....4.....C12F95B95F.C.765.A38.354D8	A6B5FF.8DAC696129F2848C9DF1D91515A1AB2BD
27	C77B181EE7F4993216D794954D1B27F47C....1F	F498B.8D352B.A8DBCAFE2.2CAF8A7972ECBE532	929EA61D8.83AE13.8C394B154861.ADD98FDF4D
	C77.A9DB435.66B121.C9F.FD98FDF4D.....	4.....C12F95B95F.C.765.A38.354D8A6B5FF.8	DAC696129F2848C9DF1D91515A1AB2BD771D7EAF
28	82CE.FDF196643374BBA5F1B35531D45F498B.8D	352B.A8DBCAFE2.2CAF8A7972ECBE532929EA61D	8.83AE13.8C394B154861.ADD98FDF4DD5278FF5
	9AB3D4ACAAFABB5516.AAA.FD5278FF54.....C1	2F95B95F.C.765.A38.354D8A6B5FF.8DAC69612	9F2848C9DF1D91515A1AB2BD771D7EAFC77B181E
29	3.8BF5E769A899578CFD54.B7B6B291.352B.A8D	BCAFE2.2CAF8A7972ECBE532929EA61D8.83AE13	.8C394B154861.ADD98FDF4DD5278FF562C2219.
	41B76EB77F6844A.4E646AF122C221512F95B95F	.C.765.A38.354D8A6B5FF.8DAC696129F2848C9	DF1D91515A1AB2BD771D7EAFC77B181E82CE.FDF
30	A945EE54.498731E6ABB5E9B6655A7A9BCAFE2.2	CAF8A7972ECBE532929EA61D8.83AE13.8C394B1	54861.ADD98FDF4DD5278FF562C2219.9CE49.1E
	84D7A943BCA7F33BA62A84E2B3712941.C.765.A	38.354D8A6B5FF.8DAC696129F2848C9DF1D9151	5A1AB2BD771D7EAFC77B181E82CE.FDF3.8BF5E7

Fig. 3. Evolution of differences with a difference `00000001` in the 15 intermediate rounds, and a state S set to zero before the 17th round (continued from Fig. 2). The final differences in S_4 and in S_{19} are unaffected by modification in the state entering the 17th rounds that map backwards to sparse differences. A state is displayed on two lines left-to-right from S_0 to S_{14}, and from S_{15} to S_{29}, in hexadecimal basis, replacing zeroes by dots for readability.

7 Conclusion

We presented an analysis of the SHA-3 candidate Fugue-256 focusing on its strongest building block, the final round function **G**, composed of five $G1$ rounds followed by 13 $G2$ rounds. We first exhibited in §3 a property of the $G1$ rounds that exploits the low density of the diffusion matrix **N**, allowing one to construct a distinguisher for the five $G1$ rounds. We then reported a detailed analysis of **G**'s properties regarding propagation of multiset properties: this study led us to a distinguisher for a slightly modified version of the full **G**. This distinguisher requires to observe the output of **G** on 256 (possibly partially unknown) inputs, and thus works in a black-box way. Indeed, what stands in the way between this distinguisher and the original **G** is a tweak of a half round, leaving a thin security margin.

§6 described a (non-black-box) distinguisher for the full, unmodified, **G**. This distinguisher cannot be directly applied to Fugue-256, because it seems difficult to determine two messages mapping the IV to two initial states of **G** found by the distinguisher. Nevertheless, it *does* affect the security guarantees of Fugue-256, for it shows that one can't rely upon the assumption that **G** has no structural flaw—in particular, no distinguisher—to argue that Fugue-256 shows some ideal behavior (for example, that it is indifferentiable from a random function[4]).

All the distinguishers presented need only negligible computation, and could be verified with the C source code of Fugue-256.

Arguably, the existence of probability-1 differentials and of nonrandomness in the state for as many as 15 and 30 rounds of **G** (which has 18 rounds in Fugue-256) raises doubts on the pseudorandomness of Fugue-256's digests (note that in its call for SHA-3 submissions NIST acknowledged the importance of pseudorandom behavior: "Hash algorithms will be evaluated against attacks or observations that (...) demonstrate some fundamental flaw in the design, such as exhibiting nonrandom behavior" [1]).

Acknowledgments

We would like to thank Charanjit Jutla for providing us the C code of the inverse **SMIX** function. We also thank Willi Meier, Thomas Peyrin, Christian Rechberger and the Fugue team for comments on a preliminar draft.

References

1. NIST: Cryptographic hash competition http://www.nist.gov/hash-competition.
2. Khovratovich, D.: Cryptanalysis of hash functions with structures. In: SAC. (2009)
3. Halevi, S., Hall, W.E., Jutla, C.S.: The hash function Fugue. Submission to NIST (updated) (2009)

[4] For example, note that the tweak of the SHA-3 candidate Keccak [10] was motivated by a structural distinguisher in 2^{1369} that invalidated the claims of its indifferentiability proof.

4. Gilbert, H., Peyrin, T.: Super-Sbox cryptanalysis: Improved attacks for AES-like permutations. Cryptology ePrint Archive, Report 2009/531 (2009)
5. Dodis, Y., Puniya, P.: On the relation between the ideal cipher and the random oracle models. In: TCC. (2006)
6. Coron, J.S., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. Cryptology ePrint Archive, Report 2008/046 (2008) Full version of [11].
7. Aumasson, J.P., Meier, W.: Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. NIST mailing list (2009)
8. Knudsen, L.R.: A key-schedule weakness in SAFER-K-64. In: CRYPTO. (1995)
9. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: FSE. (1997)
10. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Note on zero-sum distinguishers for Keccak-f. NIST mailing list (2010)
11. Coron, J.S., Patarin, J., Seurin, Y.: The random oracle model and the ideal cipher model are equivalent. In: CRYPTO. (2008)
12. Biryukov, A., Wagner, D.: Slide attacks. In: FSE. (1999)
13. Wagner, D.: A slide attack on SHA-1. Unpublished manuscript (June 2001)
14. Saarinen, M.J.O.: Cryptanalysis of block ciphers based on SHA-1 and MD5. In: FSE. (2003)
15. Gorski, M., Lucks, S., Peyrin, T.: Slide attacks on a class of hash functions. In: ASIACRYPT. (2008)

A Slide properties for G

Here we report en passant the sliding properties that we observed on $\mathbf{G}$. The *slide attack* [12] was initially applied to cryptanalyze block ciphers, and works by considering two copies of the encryption process and sliding them side by side such that their rounds are not aligned; e.g., for simplicity, sliding round $i+1$ of the first process with round i of the second. Slide attacks were first applied to hash functions in [13] and [14]. More recent results appear in [15].

Observe that all 5 $G1$ and all 13 $G2$ rounds of $\mathbf{G}$ are self-similar. Thus one can find two states S and S' such that the final state of $\mathbf{G}(S')$ equals the penultimate state of $\mathbf{G}(S)$, by returning $G1^{-5}(S)$ and $G1^{-5} \circ G2^{-1}(S)$ for some arbitrary S. Note that the above property holds for an arbitrary number of $G1$ and or $G2$ rounds in $\mathbf{G}$.

B Propagation of multiset properties

5	BBBB	BBPB	BBBB	BBBB	PPCC	PCCC	PCCP	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC
	PPCC	PCCC	PCCP	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC
5.5	????	P???	B???	B???	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC
	BBBB	BBPB	BBBB	BBBB	BBBB	PCCC	PCCP	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC
6	BBPB	BBBB	BBBB	BBBB	PCCC	PCCP	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	????
	P???	B???	B???	????	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	BBBB
7	????	????	????	????	PCCP	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	????	????
	????	????	????	????	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	BBBB	BBPB
8	????	????	????	????	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	????	????	????
	????	????	????	????	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	BBBB	BBPB	????
9	????	????	????	????	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	????	????	????	????
	????	????	????	????	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	BBBB	BBPB	????	????
10	????	????	????	????	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	????	????	????	????	????
	????	????	????	????	CCCC	CCCC	CCCC	CCCC	CCCC	CCCC	BBBB	BBPB	????	????	????
11	????	????	????	????	CCCC	CCCC	CCCC	CCCC	CCCC	????	????	????	????	????	????
	????	????	????	????	CCCC	CCCC	CCCC	CCCC	CCCC	BBBB	BBPB	????	????	????	????
12	????	????	????	????	CCCC	CCCC	CCCC	CCCC	????	????	????	????	????	????	????
	????	????	????	????	CCCC	CCCC	CCCC	CCCC	BBBB	BBPB	????	????	????	????	????
13	????	????	????	????	CCCC	CCCC	CCCC	????	????	????	????	????	????	????	????
	????	????	????	????	CCCC	CCCC	CCCC	BBBB	BBPB	????	????	????	????	????	????
14	????	????	????	????	CCCC	CCCC	????	????	????	????	????	????	????	????	????
	????	????	????	????	CCCC	CCCC	BBBB	BBPB	????	????	????	????	????	????	????
15	????	????	????	????	CCCC	????	????	????	????	????	????	????	????	????	????
	????	????	????	????	CCCC	BBBB	BBPB	????	????	????	????	????	????	????	????
16	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
	????	????	????	????	BBBB	BBPB	????	????	????	????	????	????	????	????	????
17	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
	????	????	????	????	BBPB	????	????	????	????	????	????	????	????	????	????
18	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????

Fig. 4. Evolution of multiset properties given property `P` in the first byte of the S_5 input to **G** variant (as per §4.3). Note that our analysis, focusing on the permutation property `P`, reports a property ??`P`? at round 17, while our experiments revealed that the property is in fact `BBPB`.