Improved security analysis of Fugue-256[⋆]

Praveen Gauravaram[1], Lars R. Knudsen[1], Nasour Bagheri[2], and Lei Wei[3]

[1] Department of Mathematics, Technical University of Denmark, Denmark
[2] Shahid Rajaee Teacher Training University, Iran
[3] School of Physical and Mathematical Sciences (SPMS),
Nanyang Technological University (NTU), Singapore.

Abstract. Fugue is a cryptographic hash function designed by Halevi, Hall and Jutla and was one of the fourteen hash algorithms of the second round of NIST's SHA3 hash competition. We consider Fugue-256, the 256-bit instance of Fugue. Fugue-256 updates a state of 960 bits with a *round transformation* **R** parametrized by a 32-bit message word. Twice in every state update, this transform invokes an AES like round function called **SMIX**. Fugue-256 relies on a *final transformation* **G** to output digests that look random. **G** has 18 rounds where each round invokes **SMIX** twice and finally the 960-bit output of the **G** transform is mapped with a transform τ to a 256-bit digest.

In this paper, we present some improved as well as new analytical results of Fugue-256 (with length-padding). First we improve Aumasson and Phans' integral distinguisher on the 5.5 rounds of the **G** transform to 16.5 rounds, thus showing *weak* diffusion in the **G** transform. Next we improve the designers' meet-in-the-middle preimage attack on Fugue-256 from 2^{480} time and memory to 2^{416}. Next we study the security of Fugue-256 against free-start distinguishers and free-start collisions. In this direction, we use an improved variant of the differential characteristic of the **G** transform shown by the designers to present an efficient distinguisher for the $\tau(\mathbf{G})(.)$ transform showing another *weak* diffusion property of **G**. We then extend this distinguisher to some interesting practical free-start distinguishers and free-start collisions for the length padded Fugue-256 in 2^{33} complexity. Finally, we show that free-start collision attacks on the length-padded Fugue-256 can be found in just $\mathcal{O}(1)$ *without* relying on the differential properties of the **G** transform and even *without* inverting it.
Keywords: Fugue-256, Hash function analysis, SHA-3 hash competition

1 Introduction

Fugue [6] is a cryptographic hash function designed by Halevi, Hall and Jutla and was one of the fourteen second round SHA-3 hash function candidates in the NIST's hash function competition [11, 12] . Fugue was not selected for the final round of the competition. Among all the second round candidates, so far, Fugue seems to be the hash function which has received the least amount of external analysis [13, 4]. Hence, it is significant to improve the analysis Fugue and develop a deeper understanding of this design.

The Fugue design can be viewed as an enhancement to the Grindahl hash function designed by Knudsen *et al.* [10] in which a large evolving internal state is maintained and the message words inserted into the state are processed using a *round transformation* and the complete state is maintained. After all the message words are processed, an extensive *final transformation* is applied to the state and part of its output is used as the digest. The Fugue design has two main instances called Fugue-256, denoted **F**-256 and Fugue-512, denoted **F**-512 that produce digests of sizes 256 bits and 512 bits respectively and other instances such as Fugue-224 and Fugue-384 are related to these two designs. All versions of Fugue can hash messages of lengths upto $2^{64}-1$ bits. This paper focuses on the analysis of Fugue-256.

F-256 maintains an internal state S of 960 bits as a 4×30 matrix with each column containing a 32-bit word S_i for $i = 0, \ldots, 29$. It updates S by using a *round transformation* **R** parametrized by a message word of 32 bits. Once all message words are processed, the state undergoes through a *final transformation* **G**, composed of 5 rounds of **G1** and 13 rounds of **G2**. Finally, a transform τ maps the output state to a digest of size eight words and the composition of τ and **G** is denoted by $\tau(\mathbf{G}(.))$. The main component of **F**-256 is a 16-byte to 16-byte permutation transformation called **SMIX** which consists of a substitution box **SBox** followed by a linear transformation **SMIX-T** resembling the round functions of the AES block cipher [3].

[⋆] A short version of this paper will be published in the Lecture Notes in Computer Science (LNCS) proceedings of Australasian Conference on Information Security and Privacy (ACISP), 2011, Springer.

By using proof-oriented methods, the designers proved that mounting a collision attack on **F**-256 would require a work factor of at least 2^{128} and finding a (second) preimage requires a work factor of 2^{256} [6, § 1.2]. They also claimed that since the *round transformation* and *final transformation* of **F**-256 are invertible, a generic meet in the middle attack can be applied to **F**-256 to find a preimage in 2^{480}. Khovratovich [8] showed a collision attack on the internal states of **F**-256 and **F**-512 with time and memory complexities of 2^{352} and 2^{480} respectively. Aumasson and Phan [2, 1] showed an efficient distinguisher on the transform $\tau(\mathbf{G}(.))$ of **F**-256 which distinguishes it from a random function. In this attack, they showed pairs of input states to the **G** transform that differ on average in only 66 bits such that their digests are the same for all pairs found. They also showed an integral distinguisher on the 5.5 rounds of the **G** transform and on a tweaked variant of the **G** transform. Turan and Uyan [14] showed practical semi-free start collisions and semi-free start near-collisions by using hill-climbing techniques for the reduced versions of **F**-256 where the final **R** transform had reduced rounds.

In this paper, we present improved analytical results on **F**-256 building upon the previous analysis considered by the designers [6] and Aumasson and Phan [2, 1]. First we improve the known integral distinguisher [2] on the 5.5 rounds of the **G** transform of **F**-256 to 16.5 rounds. This attack shows that **G** transform has *weak* diffusion. We then improve the designers' generic meet-in-the-middle preimage attack on any instance of Fugue with n-bit internal state from a complexity of $2^{n/2}$ to $2^{n/2-16}$. We further improve this attack on **F**-256 by a factor of 2^{48} by exploiting the freedom available in the message words and in some state words. Next we consider the problem of doing free-start distinguisher and free-start collision attacks for Fugue-256. In this direction, we first present an efficient distinguisher for the $\tau(\mathbf{G}(.))$ transform by using an improved variant of the differential characteristic for the **G** transform considered by the designers in the PRF analysis of **F**-256. This distinguisher when extended to **F**-256 shows an interesting free-start distinguisher for **F**-256 in 2^{33} work factor. We note that similarly the distinguisher of the Aumasson and Phan on the $\tau(\mathbf{G}(.))$ transform can be extended to a free-start distinguisher for **F**-256 for a similar complexity with some subtle changes. These distinguishers also produce free-start collisions for no additional cost. Finally, we show that it is indeed "trivial" to show much efficient free-start collisions for the length-padded Fugue-256 in $\mathcal{O}(1)$ *without* relying on the differential characteristics of the **G** transform and even *without* inverting it. We remark that this type of free-start collision attack is different from the well known trivial free-start collision attack on **F**-256 that requires inverting **R** and **G** transforms twice for the unpadded **F**-256.

The paper is organised as follows: In §2 and §3, the description of **F**-256 and some notation used in the paper are provided. Analysis is presented in §4, §5, §6, §7 followed by the conclusion in §8.

2 F-256 hash function

F-256 parses the 256-bit initial value (IV) as eight 4-byte words $IV_0, \ldots, IV_7$. It initializes a state S of 30 4-byte words S_i for $i = 0, \ldots, 29$, as a 4×30 matrix by assigning $S_j = 0$ for $j \in [0, 21]$ and $S_j = IV_{j-22}$ for $j \in [22, 29]$. This state of **F**-256 is called *initial state*. Hereafter, we denote by $S_{i\sim j}$ the consecutive words of a state S from the index i to j (including i and j). Streams of 4-byte message word inputs are processed from this state using a *round transformation* **R**. If the input message is not a multiple of 32 then **F**-256 pads the message with sufficient 0 bits so that the padded message is a multiple of 32. The padded message is appended with an additional 64 bits (i.e two 4-byte words) that represent the binary encoding of the length of the unpadded message in big-endian notation. Once the length encoded message is processed with the **R** transform, a *final transformation* **G** is applied to the internal state to obtain an *output state* of 30 words. The eight words $S_{1\sim4}$,$S_{15\sim18}$ of the *output state* are used as the digest. The transforms **R** and **G** are discussed below where the addition $+$ is addition of vectors of four bytes in GF(2^8), and hence is the same as 32-bit exclusive-or and for 4-byte vectors a and b, $a+=b$ means $a = a + b$.

Round transformation (R). The **R** transform takes a state S and a 4-byte message word m as inputs and outputs a new thirty column state. The transformation **R** calls a sequence of functions: **TIX**(m), **ROR3**, **CMIX**, **SMIX**, **ROR3**, **CMIX**, **SMIX**.

- The function **TIX**(m) has the following steps:
  - $S_{10}+= S_0$; $S_0 = m$

  - $S_8 += S_0$; $S_1 += S_{24}$
- The function **ROR3** rotates the state to the right by three columns, that is $S_i = S_{i-3 \bmod 30}$.
- The column mix function **CMIX** has the following steps:
  - $S_0 += S_4$; $S_1 += S_5$; $S_2 += S_6$
  - $S_{15} += S_4$; $S_{16} += S_5$; $S_{17} += S_6$
- The **SMIX** transform operates only on the first four columns $S_{0\sim3}$ of the state S that are viewed as a 4×4 matrix of 16 words. Each byte of these columns first undergoes an **SBox** transform which is the one as in AES [3] and the resulting matrix undergoes an **SMIX-T** transform denoted by a 16×16 matrix **N** of 256 bytes. That is, $S'_{0\sim3} = \mathbf{N}.(S_{0\sim3})$ where **N** is multiplied (.) with a 16-byte 4×1 column matrix output of **SBox**. Similarly, $(S_{0\sim3}) = \overline{\mathbf{N}}.(S'_{0\sim3})$ where $(S'_{0\sim3})$ is a 16-byte 4×1 column matrix.

Final transformation (G). The **G** transform takes the output S of the **R** transform and produces a *final state* of 30 words. The function **G** consists of 5 rounds of **G1**, 13 rounds of **G2** and a binary addition of two state words. The five rounds of **G1** are denoted by $\mathbf{G1_1}, \ldots, \mathbf{G1_5}$ and thirteen rounds of **G2** are denoted by $\mathbf{G2_1}, \ldots, \mathbf{G2_{13}}$. These operations on a state S are given by:

- Function **G1**: It is a sequence of functions: **ROR3**, **CMIX**, **SMIX**, **ROR3**, **CMIX**, **SMIX**.
- Function **G2**: It has the following steps:
  - $S_4 += S_0$; $S_{15} += S_0$; **ROR15**; **SMIX**
  - $S_4 += S_0$; $S_{16} += S_0$; **ROR14**; **SMIX**
- $S_4 += S_0$; $S_{15} += S_0$

In the above, **ROR15** and **ROR14** mean rotations of the state S to the right by 15 and 14 columns respectively. The resultant state is called *final state*.

256-bit digest. After discarding the words S_0,$S_{5\sim14}$ and $S_{19\sim29}$ from the *final state*, the concatenation of the words $S_{1\sim4}$ and $S_{15\sim18}$ is used as the digest. This step of producing the digest from the **G** transform is denoted by $\tau(\mathbf{G}(S))$ where S is the input state of **G**.

3 Notation

In this section, we introduce some notation on **F**-256 and definitions that are used later in our analysis. Notation specific to some parts of the analysis will be introduced in the relevant sections.

In any round i of **R**, the internal state (also the starting state of Round i) is denoted by *State-i* and its words are denoted by $S_0^i, S_1^i, \ldots, S_{29}^i$, i.e, $S_{1\sim29}^i$. The internal state words after the first **SMIX** in a round i are denoted by $S_0^{i.5}, \ldots, S_{29}^{i.5}$, i.e, $S_{0\sim29}^{i.5}$. In any round i of **R**, the internal state words after the first **ROR3**, **CMIX** and **SBox** transformations are denoted by $x_0'^i, \ldots, x_{29}'^i$, $x_0^i, \ldots, x_{29}^i$ and $\hat{x}_0^i, \ldots, \hat{x}_{29}^i$ respectively and those after the second **ROR3**, **CMIX** and **SBox** transformations are denoted by $y_0'^i, \ldots, y_{29}'^i$, $y_0^i, \ldots, y_{29}^i$ and $\hat{y}_0^i, \ldots, \hat{y}_{29}^i$ respectively. We indicate any non-zero difference in the state words by placing δ before those words. For example, differences in the words of *State-i* are denoted by $\delta S_0^i, \ldots, \delta S_{29}^i$, i.e, $\delta S_{0\sim29}^i$. A message word inserted in the i^{th} round of **R** is denoted by m^i. Message words in the rounds from i to j are denoted by $m^i \sim m^j$ and differences in the message words in these rounds by $\delta m^i \sim \delta m^j$.

4 Integral distinguisher for the 16.5 rounds of the G transform of F-256

Our integral attack is a first order integral attack. We follow the notation of [9] for the bytes included in the integral as follows: The symbol $\mathcal{C}$ (for Constant) in the i^{th} byte means that the values of all i^{th} bytes in the attack are equal. The symbol $\mathcal{A}$ (for All) means that all bytes in the attack are different, and the symbol $\mathcal{S}$ (for Sum) means that the sum of all i^{th} bytes is predictable and we write ? when the sum of the bytes is not predictable. We count the rounds of the **G** transform from 0 to 17 and a state in any round i where $i = 0, 0.5, 1, \ldots, 16, 16.5, 17$ is denoted by S^i and the words of S^i by $S_{0\sim29}^i$.

Integral distinguisher on the 5.5 rounds of the G transform [2]. Aumasson and Phan [2] presented an integral distinguisher for 5.5 rounds of the **G** function. Their distinguisher fixes all the bytes of the state S^0 except for the first byte of S_2^0 at the start of the **G** transform. All possible values are assigned to the first byte of S_2^0. They have shown that for $S_2^0 = \mathcal{A}\|\mathcal{C}\|\mathcal{C}\|\mathcal{C}$ one would receive $S_0^{5.5} = ?\|?\|?\|?, S_1^{5.5} = \mathcal{A}\|?\|?\|?, S_2^{5.5} = \mathcal{S}\|?\|?\|?, S_3^{5.5} = \mathcal{S}\|?\|?\|?$ which presumably shows a non-randomness property in the first 5.5 rounds of the **G** function. In addition, this attack was extended to a tweaked version of **G**-function. We improve their attack on 5.5 rounds of **G**-function such that it can be applied to 16.5 rounds out of 18 rounds.

4.1 Improved attack

A closer analysis of integrals reveals that the values of the integral before the **ROR** functions of **G2** play a crucial role on the success of the distinguisher. It turns out that this word remains unchanged through many rounds of **G2** before being affected by other words. However, for the given integral, all bytes of $S_0^{5.5}$ are unknown ('?') and out of control of the adversary. Hence, the integral of Aumasson and Phan does not seem to extend to more rounds of the **G** transform. Our analysis revealed an integral that runs for more rounds. The propagation of our integral has been depicted in Table 3 in Appendix B. It should be mentioned that values with notation $\mathcal{A}$ and $\mathcal{C}$ in the integral are unchanged through **SBox**, but values with notation $\mathcal{S}$ are unknown (?) after **SBox**.

In our integral, we fix whole state bytes of S^0, except for the second byte of S_4^0 where we consider all possible values. The word S_4^0 propagates to S_{28}^5 with probability 1. Hence, the **ROR3** transform in the 5th round of **G1** (i.e 4th round of **G**) shifts this word as one of the inputs to the **SMIX**. Hence, we obtain $S_0^{5.5} = ?\|?\|\mathcal{S}\|?$. It means that we know the sum of the values ($\mathcal{S}$) for this word. On the other hand, this word is propagated to S_4^{16} with probability 1.

In the next step, we have $S_4^{16} + = S_0^{16}$ which destroys our integral. However, after the **ROR15** function in the 16th round of **G**, S_0^{16} and S_4^{16} are propagated to S_{15}^{16} and S_{19}^{16} respectively. Now if we assume that the adversary has also access to S_{19}^{16} then he can combine S_{15}^{16} and S_{19}^{16} and retrieve the integral values as $S_4^{16.5} = S_{15}^{16.5} \oplus S_{19}^{16.5}$.

Hence, we have an integral which applies to 16.5 out of 18 rounds of the **G** transform. The computational complexity of attack is 256 evaluations of 16.5 rounds of the **G** transform and memory is 256 bytes. The probability of receiving an $\mathcal{S}$ byte at $S_4^{16.5}$ for **G** is 1 whereas this probability for a random permutation is 2^{-8}. Hence the success probability to distinguish 16.5 rounds of the **G** function from a random permutation is $1 - 2^{-8}$. Our findings illustrate the weak diffusion of the **G** transform.

5 Improved meet-in-the-middle preimage attacks on Fugue

5.1 Generic meet-in-the-middle preimage attack on Fugue [6, p.77]

The designers of Fugue noted the application of a generic meet-in-the-middle preimage attack on any t-bit instance of Fugue [6, p.77] (along with length padding) with n-bit ($n/32$-word) internal state in $2^{n/2}$ time and memory complexity. For a given digest Y, this attack finds a preimage for Fugue as follows:

1. *Forward process:* Choose $2^{n/2}$ messages of equal length and process them along with their 64-bit length padding to the corresponding internal states at *State-i* of some round i of **R** from the *initial state* of Fugue. Store these messages and their internal states at *State-i* in a Table L_1.
2. *Backward process:* Fill the output state words $S_{1\sim 4}$ and $S_{15\sim 18}$ with the digest Y. Build $2^{n/2}$ *final states* by choosing random values for the remaining 22 words and invert the *final transformation* **G** from these states to build $2^{n/2}$ internal states at *State-i*. Store these state values in a Table L_2.
3. Due to birthday paradox, an internal state in L_1 collides with an internal state in L_2 with a good probability and this event is called a *collision match*. The internal state *State-i* at which the *collision match* occurs is called the *middle state*. Finally, produce the concatenation of message blocks excluding the two length padding blocks from the *Forward process* that contributed to the *collision match* as the preimage of Y. Note that this attack generates a preimage of size at least $(n/2)/32$ words.

In summary, the complexity of this preimage attack is influenced by the internal state size. On **F**-256 and **F**-512, the time and memory complexities of this attack are 2^{480} and 2^{576} respectively.

5.2 Improved generic meet-in-the-middle preimage attack on Fugue

Let *State-i'* be the internal state in any round i of the $\mathbf{R}$ transform after the step $S_{10}^i = S_0^i \oplus S_{10}^i$. This is a $(n-32)/32$-word internal state without the word S_0^i and except for the word S_{10}^i in the *State-i'* all other words are the same as in the $n/32$-word *State-i*. Instead of looking for a *collision match* at *State-i* in the generic MIM attack, if we look for it at *State-i'*, we can improve the generic attack complexity by a factor of 2^{16}. The improved attack works as follows:

1. *Forward process:* This step is similar to the *Forward process* step of the generic MIM attack except that length padding message words are not processed and $2^{n/2-16}$ messages are processed till the internal states at *State-i'* of some round i of $\mathbf{R}$. Finally, $2^{n/2-16}$ messages and corresponding internal states at *State-i'* are stored in a Table L_1.
2. *Backward process:* The following steps are executed as part of this process:
   (a) Establish the valid length-padding for the attack:
      – Fill the words $S_{1\sim4}$ and $S_{15\sim18}$ of the *final state* with the digest Y.
      – Use the freedom available in the remaining $n-t$ words of the *final state* and invert the *final state* until the desired length-padding word in the first round of $\mathbf{R}$ (in the *backward* direction) after the $\mathbf{G}$ transform is obtained. This requires about 2^{32} inversions for the $\mathbf{G}$ transform.
      – To obtain the desired length-padding word in the second round of $\mathbf{R}$ (in the *backward* direction), we use freedom in the words S_0 and S_{10} of this round of $\mathbf{R}$ and follow the steps 2 and 3 of the structural pseudo differentiator of $\mathbf{F}$-256 explained in §6.3.
   (b) Generate required number of messages with length-padding and internal states:
      – Let j and $j-1$ be the rounds of the $\mathbf{R}$ transforms where we have obtained the two valid length-padding words. Now, we also have obtained a fixed $(n-32)/32$-word state *State-(j-1)'* at round $j-1$ where only the word S_0^{j-1} is not fixed. We use 32-bit freedom in this word combined with the word S_{10}^{j-1} which is xored with S_0^{j-1} to generate 2^{32} $n/32$-word internal states *State-(j-1)*. By inverting these 2^{32} internal states through the round $j-2$ we obtain 2^{32} internal states and message words at round $j-2$ of the $\mathbf{R}$ transform.
      – We repeat the above step for every fixed $(n-32)/32$-word state in every round by using the freedom in the word S_0 for a sufficient number of rounds until we generate $2^{n/2-16}$ messages and corresponding internal states. That is, we need at least $(n/2-16)/32$ $\mathbf{R}$ transform inversions from the round $j-1$ and $((n/2-16)/32)+2$ in total to account for length padding. This means that the size of each of $2^{n/2-16}$ messages is $((n/2-16)/32)+2$ words. All these internal states and messages are stored in a Table L_2. For instance, $\mathbf{F}$-256 requires a total at least 17 rounds of $\mathbf{R}$ inversions and $\mathbf{F}$-512 requires at least 20 such rounds.
   (c) The complexity of this process is about $2^{n/2-16}$ operations of the $\mathbf{R}$ transform and similar memory and 2^{32} operations of the $\mathbf{G}$ transform.
3. Find a *collision match* at the *middle state* between the internal states in the tables L_1 and L_2. The complexity of this improved attack is on average $2^{n/2-16}$ time and memory and generates a preimage of size at least twice of the one due to the generic MIM preimage attack.

As an illustration, the improved attack finds a preimage of size at least 29 (resp. 35) message words with a complexity of 2^{464} (resp.2^{560}) time and memory for $\mathbf{F}$-256 (resp. $\mathbf{F}$-512). We remark that these preimage attack bounds are the same as the trivial internal collision attack on these instances of Fugue pointed out by Khovratovich [8].

5.3 Improved meet-in-the-middle preimage attack on F-256

We further improve the preimage attack on $\mathbf{F}$-256 from §5.2 by exerting control over 3 words of the 29-word *middle state*. This technique allows us to use a birthday attack to match only 26 words of the *middle state*, thereby reducing the complexity of the attack to 2^{416} from 2^{464}. For this purpose, we reduce the number of $\mathbf{R}$ transforms required in both the *Forward process* and *Backward process* of the attack in §5.2 by 1.5 rounds and control 3 rounds on either side of the *middle state* deterministically. Let 0 be the round of the $\mathbf{R}$ transform at which we aim for a *collision match*. Let $-1, -2, -3, \ldots$ and $1, 2, 3, \ldots$ be the respective rounds of the $\mathbf{R}$ transforms from the 0th round in the *Forward process* and *Backward process* of the attack. The attack is outlined below:

1. We show that the words S^0_{17}, S^0_{23} and S^0_{27} in the *middle state* (i.e *State-0'*) can be controlled such that the internal states evolving from the *initial state* and the *final state* of **F**-256 can be matched in these words deterministically with a probability of 1 by solving a simple system of equations. We do this by first assigning fixed values[4] to the words S^0_{17}, S^0_{23} and S^0_{27} that are controlled by using the **R** transforms -3, -2 and -1 in the *Forward process* and the **R** transforms 3, 2 and 1 in the *Backward process*. In the *Forward process*, the desired value for the words S^0_{27}, S^0_{23} and S^0_{17} is obtained consecutively by using the freedom available in the message words m^{-3}, m^{-2} and m^{-1} in the **R** transforms of the rounds -3, -2 and -1 respectively. In the *Backward process*, the desired values for the words S^0_{17}, S^0_{23} and S^0_{27} are obtained consecutively by using the freedom available in the words S^3_0, S^2_0 and S^1_0 in the **R** transforms of the rounds 3, 2 and 1 respectively. Below we explain how the word S^0_{17} can be controlled and a similar explanation follows for controlling the words S^0_{23} and S^0_{27}.
   (a) *Controlling the word S^0_{17}:* Below we will show how we can obtain the desired word S^0_{17} of the *middle state* from the *final state* and *initial state* of **F**-256 through the *Backward process* and *Forward process* respectively.
      i. *Backward process:* The word S^0_{17} in the *middle state* of the 0th round **R** transform will be the word S^2_{29} in the *State-2'* of the 2nd round **R** transform. Now $x'^2_2 = S_{17^0}$, $x^2_2 = x'^2_2 \oplus x'^2_6$. Now $\hat{x}^2_2 = \mathbf{SBox}(x^2_2)$. Note that $(S^{2.5}_1, S^{2.5}_2, S^{2.5}_3) = (S^3_4, S^3_5, S^3_6)$. For the *final state* of **F**-256 inverted till the **R** transform in round 3 by following the *Backward process* algorithm in §5.2, the *State-3'* of the 3rd round **R** transform is fixed. Therefore, we can only use $S^{2.5}_0$ input to $\overline{\mathbf{N}}$ to obtain the desired $\hat{x}^2_2$ and therefore, we can obtain the desired S^0_{17}. The matrix **N** has a property that by controlling one of the input words, we can obtain one desired output word by solving a system of 4 equations in 4 unknowns for a negligible complexity. This property is also applicable for $\overline{\mathbf{N}}$. Hence, we can find a $S^{2.5}_0$ such that $\overline{\mathbf{N}}.(S^{2.5}_0, S^{2.5}_1, S^{2.5}_2, S^{2.5}_3)$ produces the desired word $\hat{x}^2_2$. This process also determines the message word m^2 which is $\overline{\mathbf{SBox}}(\hat{x}^2_3) = x^2_3 = x'^2_3$. Note that $S^{2.5}_0 = y'^2_3 = y^2_3$, $\hat{y}^2_3 = \mathbf{SBox}(y^2_3)$ and the state words $S^3_{1\sim 29}$ of the state *State-3'* are determined by the *final state* of **F**-256. Now we vary S^3_0 such that $\overline{\mathbf{N}}.(S^3_0, S^3_1, S^3_2, S^3_3)$ produces the desired $\hat{y}^2_3$. Once we have found the candidate S^3_0, we can determine $S^3_{10} = S^3_0 \oplus x'^3_{13}$ of the state *State-3*.
      ii. *Forward process:* For an *initial state* of **F**-256 processed till the end of the -2nd **R** transform by using the *Forward process* algorithm of §5.2, the *State-(-1)* of the -1th round **R** transform is fixed. This implies that y'^{-1}_{17} has already been fixed. To obtain the desired value of S^0_{17}, we need to control y'^{-1}_6 which is $y'^{-1}_{17} \oplus S^0_{17}$. The word y'^{-1}_6 is the same as $S^{-1.5}_3$. Note that the words $S^{-1}_{1\sim 29}$ had already been fixed. Hence, we can determine the words ($\hat{x}^{-1}_0$, $\hat{x}^{-1}_1$, $\hat{x}^{-1}_2$), the first three word input to **N**, as follows:
         A. $\hat{x}^{-1}_0 = \mathbf{SBox}(S^{-1}_{27} \oplus S^{-1}_1 \oplus S^{-1}_{24})$
         B. $\hat{x}^{-1}_1 = \mathbf{SBox}(S^{-1}_{28} \oplus S^{-1}_2)$
         C. $\hat{x}^{-1}_2 = \mathbf{SBox}(S^{-1}_{29} \oplus S^{-1}_3)$

         Having determined the words $\hat{x}^{-1}_0$, $\hat{x}^{-1}_1$ and $\hat{x}^{-1}_2$, we can use the freedom available in the message word m^{-1} to determine the candidate $\hat{x}^{-1}_3$ such that we obtain the desired $S^{-1.5}_3 = y^{-1}_6$ and therefore, we obtain the desired word $S^0_{17} = y^{-1}_6 \oplus y'^{-1}_{17}$ in the *middle state.*

Note that this attack produces a preimage for **F**-256 of size at least 32 words excluding the length-padding words.

Remark 1. It is difficult to exert control over more than 3 words in the *middle state* of the 0th round **R** transform deterministically with a probability of 1 as the message word m^{-3} from the -3rd round **R** transform influences the starting internal state of the 0th round **R** transform.

6 A free-start distinguisher for F-256 (with length padding)

Designers of Fugue considered a differential characteristic in the analysis of the PRF mode of **F**-256 [6, §12.4.2]. This characteristic was not fully specified in the documentation although the designers claim that this path does not lead to any *active* SMIX-es in many rounds of **G** without any proof. This path

[4] These three words can have different values but they must be fixed.

via some $\mathbf{R}$ transforms was used to prove that the probability of obtaining partial collision on the internal state (at the start of the $\mathbf{G}$ transform) for the $\mathbf{F}$-256 PRF is at most 2^{-142}. The designers also noted that this argument applies for a known or chosen key/Initial Value (IV) model of $\mathbf{F}$-256 [5]. However, no analysis was provided by the designers for the case where an attacker can have free start initial values leading to a free-start distinguisher for $\mathbf{F}$-256 and see whether it is possible to obtain partial collisions on the internal state for this case.

In this section by using an improved differential characteristic of the designers we show a distinguisher for the $\tau(\mathbf{G}(.))$ transform in much similar style as done by Aumasson and Phan [2, 1]. Then we extend it to a free-start distinguisher on $\mathbf{F}$-256 that result in a pair of chosen IVs (or free IVs) that are close to each other on average in 332 bits of 960-bit initial state producing partial collisions on the internal state where 80% of the internal state has a collision. Our findings, however, do not contradict the designers' arguments for the known or chosen key/Initial Value (IV) model of $\mathbf{F}$-256 [5]. However, they complement the analysis of the designers by considering free-start initial value case.

To make presentation easier, we show our analysis by developing the differential characteristic from scratch considering that a variant of it was not specified in the design documentation of Fugue [6]. Later in the section we point out the difference between our path and the one considered by the designers. We present in §6.1 and §6.2 the tools required to do this attack which is explained in §6.3.

6.1 Differential property of SMIX-T

Recall that **SMIX-T** is a linear transform represented as a 16×16 matrix $\mathbf{N}$ and $\mathbf{N}.(S_{0\sim3}) = (S'_{0\sim3})$ and therefore, $\overline{\mathbf{N}}.(S'_{0\sim3}) = (S_{0\sim3})$ where $(S_{0\sim3})$ and $(S'_{0\sim3})$ are 4×1 column matrices. We note that it is easy to find an input difference $(\delta S'_0, \delta S'_1, \delta S'_2, \delta S'_3)$ to $\overline{\mathbf{N}}$ such that we get an output difference $(\delta S_0, \delta S_1, \delta S_2, \delta S_3)$ where $\delta S'_0 = 0$, $\delta S'_3 = 0$ and $\delta S_3 = 0$. That is, we can find a pair of inputs (S'_0, S'_1, S'_2, S'_3) and $(S'_0, S'^*_1, S'^*_2, S'_3)$ to $\overline{\mathbf{N}}$ that collide on the word S_3 for any S'_0 and S'_3 where $\delta S'_1 = S'_1 \oplus S'^*_1$ and $\delta S'_2 = S'_2 \oplus S'^*_2$. This attack can be precomputed and the solution for this attack is the solution to 8 linear equations in 8 unknowns which requires negligible computational cost. We can also fix 3 bytes in one of the differences of $\delta S'_1$ and $\delta S'_2$ to zeroes which leads to solving 5 linear equations in 5 unknowns. As shown later in our attack, the difference $\delta S'_1$ propagates to more words in the input state of the $\mathbf{R}$ transform compared to $\delta S'_2$. Hence, to minimise the number of *active* state input bits of our attack, we fix 3 bytes of $\delta S'_1$ to a zero difference and vary the other word for a solution. By running an experiment for a minimum weight solution, we found $\delta S'_1 = 0\text{x } 00000009$ and $\delta S'_2 = 0\text{x } 5042d427$. These differences are *fixed* and we call them δ_1 and δ_2 respectively.

6.2 Efficient distinguisher for the $\tau(\mathbf{G}(S))$ transform

We can choose any intermediate state in the $\mathbf{G}$ transform and proceed *forwards* and *backwards* to compute its corresponding *final* and input states. This is called *inside-out* strategy [1] which we use to distinguish the $\tau(\mathbf{G}(S))$ transform from a random function with a probability of 1.

Forwards from $\mathbf{G2}_{11}$. We choose an internal state at the start of the round $\mathbf{G2}_{11}$ such that the words S_{18} and S_{19} have the differences δ_1 and δ_2 computed in § 6.1 and the remaining words ($S_{0\sim17}$ and $S_{20\sim29}$) have the zero difference. The differences δ_1 and δ_2 activate **SMIX**-es in the rounds $\mathbf{G2}_{11}$ and $\mathbf{G2}_{12}$ respectively. The digest returned after the $\mathbf{G}$ transform depends on the differences δ_1 and δ_2 but not on all the state words at the start of the round $\mathbf{G2}_{11}$. This observation allows us to find distinct pairs of states at $\mathbf{G2}_{11}$ such that the difference of their respective digests is fixed for all pairs found. A closer analysis shows that the digest does not depend on the state words $S_{8\sim14}$ and $S_{22\sim29}$ at the start of the round $\mathbf{G2}_{11}$. Hence, in the round $\mathbf{G2}_{11}$ we can build many pairs of states by fixing the words $S_{0\sim7}$, $S_{15\sim17}$ and $S_{20\sim21}$ with the same actual value, the words S_{18} and S_{19} with some values that differ by δ_1 and δ_2 respectively and varying the remaining words with zero difference for all pairs of states to obtain digests that have the fixed difference.

Backwards from $\mathbf{G2}_{11}$. Consider any two internal states at the start of $\mathbf{G2}_{11}$ that satisfy the above constraint of having the words $S_{0\sim7}$, $S_{15\sim17}$ and $S_{20\sim21}$ fixed with the same value (i.e, zero difference) and the words S_{18} and S_{19} fixed to some value but with the differences δ_1 and δ_2 respectively. When we process these two states until the end of the round $\mathbf{G1}_2$ in the backward direction, we get two intermediate states with the differences δ_1 and δ_2 appearing in the words S_4 and S_5. All other words have zero differences. When the second half of $\mathbf{G1}_1$ is inverted, we obtain the difference δ_1 (resp. δ_2) in the words S_1, S_{12} and S_{27} (resp. S_2, S_{13} and S_{28}). When we invert the remaining half of $\mathbf{G1}_1$, the differences δ_1 and δ_2 in the state words S_1 and S_2 activate $\overline{\mathbf{SMIX}}$ creating uncontrolled differences in the words $S_{0\sim2}$ and zero difference in the word S_3 as shown in § 6.1. Hence, we get the input state to the $\mathbf{G}$ transform with the difference δ_1 (resp. δ_2) in the words S_9 and S_{24} (resp. S_{10} and S_{25}) and uncontrolled differences in the words $S_{27\sim29}$. That is, only seven words of the input state to the $\mathbf{G}$ transform have differences. Recall that the differences δ_1 and δ_2 are not chosen arbitrarily and they are determined by the attack described in § 6.1. For our solution, δ_1 and δ_2 have the weights of 2 and 12 respectively, and the uncontrolled differences in the words $S_{27\sim29}$ have, on average, a weight of 48.

Hence, we have shown a method which finds pairs of state inputs (S, S^*) to the $\mathbf{G}$ transform that differ, on average, in 76 bits such that the difference of $\tau(\mathbf{G}(S))$ and $\tau(\mathbf{G}(S^*))$ remains *fixed* for all such pairs found.

6.3 Free-start distinguisher for F-256

In this section, we extend distinguisher on the $\tau(\mathbf{G}(S))$ transform to a free-start distinguisher on **F**-256 hash function. Recall that **F**-256 uses a 64-bit length padding in the last two rounds of $\mathbf{R}$ transform representing the binary encoded form of the message. Hence, we show our attack on the composition of 3 rounds of $\mathbf{R}$ (last 2 rounds are used for the length padding) and $\tau(\mathbf{G}(S))$. Note that $0, -1, -2, \ldots$ are the rounds of $\mathbf{R}$ before the $\mathbf{G}$ transform and $m^0, m^{-1}, \ldots$ are their respective message blocks. Recall that *State-i'* is the 29-word internal state (from word 1 to 29) in any round i of the $\mathbf{R}$ transform after the step $S^i_{10} = S^i_0 \oplus S^i_{10}$ in the *forward process*. We denote a pair of states at *State-i'* of $\mathbf{R}$ by $(S[i], S[i]')$ where $S[i] = S[i]_{1\sim29}$ and $S[i]' = S[i]'_{1\sim29}$. The attack is outlined below:

1. Use the freedom available in the state words $S_{8\sim14}$ and $S_{22\sim29}$ in Round $\mathbf{G2}_{11}$ of the distinguisher for the $\tau(\mathbf{G}(S))$ transform to find a pair of states $(S[0], S[0]')$ such that their common message block is $m^0 = 0x00000020$, which is the last length encoded word of a 32-bit message. The cost of this step is 2^{32} calls to the **Backwards from $\mathbf{G2}_{11}$** step of the distinguisher for the $\tau(\mathbf{G}(S))$ transform. Note that by now the *actual values* of the words $S[0]_{1\sim29}$ and $S[0]'_{1\sim29}$ in these pair of states are fixed. This implies that the *actual values* of the words $S^0_{1\sim9}$ and $S^0_{11\sim29}$ are also fixed and the values of the words S^0_0 and S^0_{10} are not fixed.
2. Now we use the freedom in the word S^0_0 to find a pair of states (S^{-1}, S'^{-1}) that have the same message block $m^{-1} = 0x00000000$ in the -1th $\mathbf{R}$ transform. We do this as follows:
   (a) The word S^0_4 which has the difference δ_2 will also be the 1st word input to the second $\overline{\mathbf{SMIX}}$ transform (in the *backward process*). This $\overline{\mathbf{SMIX}}$ transform also has the non-zero difference at input word 0 which is the same as the output of word 3 of the first $\overline{\mathbf{SMIX}}$ transform of -1th round of the $\mathbf{R}$ transform. Moreover, the actual values of the input words 2 and 3 to the second $\overline{\mathbf{SMIX}}$ transform are already fixed and they are zero difference words. Now we find the *actual values* of the input word 0 of the pair of states for the second $\overline{\mathbf{SMIX}}$ transform such that we obtain the 3rd output word of this second $\overline{\mathbf{SMIX}}$ as $0x00000000$. This can be done by solving a simple system of linear equations.
   (b) Having determined the *actual value* of the input word 0 to the second $\overline{\mathbf{SMIX}}$ transform for the pair of states of -1th round of the $\mathbf{R}$ transform, we can use the freedom in the input word S^0_0 of the first $\overline{\mathbf{SMIX}}$ transform of -1th round of the $\mathbf{R}$ transform to force its output 3rd word to the *actual value* computed in the above step 2(a). This can be done by solving a simple system of linear equations. Note that the word S^0_3 has δ_1 difference and the words $S^0_{1\sim2}$ have zero difference with their *actual values* already determined.
3. Having obtained a pair of 29-word states $(S[-1], S[-1]')$ with the common message $m^{-1} = 0x00000000$, we choose pairs $(S^1_0, S^{1'}_0)$ and $(S^1_{10}, S^{1'}_{10})$ such that $S^1_0 \oplus S^1_{10} = S[-1]_{10}$ and $S^{1'}_0 \oplus S^{1'}_{10} = S[-1]'_{10}$. Note that since $\delta S^1_{10} = 0$, we have $S^1_{10} = S^{1'}_{10}$ and $S^1_0 = S^{1'}_0$. Now we have determined a pair of 30-word states $(S^1, S^{1'})$ at the start of the -1th $\mathbf{R}$ transform.
4. Invert the 30-word states $(S^1, S^{1'})$ for one more round of the $\mathbf{R}$ transform and obtain the states (S^{-2}, S'^{-2}) and corresponding message blocks m^{-2} of the -2th $\mathbf{R}$ transform.

By repeating this attack using the freedom available in the state words in the round $\mathbf{G2}_{11}$, we can find another pair of *free-start initial states* (S^{-2}, S'^{-2}) together with the corresponding message blocks m^{-2} such that they have the same digest difference as the first pair of states and their respective messages. Hence, the complexity of the attack is 2^{33} hashing operations and requires negligible memory. These pairs of *free-start initial states* have a symmetrical structure at the input that their difference in twelve words[5] is the same and they differ on average in only 332 bits of the 960-bit *input state*, thus illustrating *weak* state diffusion in **F**-256. In addition, they also result in free-start collisions for no additional cost. The differential characteristic and examples which illustrate this distinguisher and free-start collisions are shown in Appendix A.

6.4 Extending the distinguisher to more R transforms.

When the free-start distinguisher on 3 rounds of $\mathbf{R} \circ \mathbf{G}$ of **F**-256 is extended by one more round of the **R** transform, symmetrical pattern in the pairs of *free-start input states* that produce the same digest difference can be observed in 6 words ($S_{6\sim11}$ of which 4 have 0 difference). In this case, the actual value of the length padding accounts for a 2-word message but the cost of the attack remains the same as 2^{33} hashing operations. Similarly, for 5 rounds of $\mathbf{R} \circ \mathbf{G}$ of **F**-256, symmetrical pattern in the pairs of *free-start initial states* can be observed in 4 words ($S_{3\sim5}$ having 0 difference) and complexity of the attack is unchanged. For 6 rounds of **R**, the whole state gets diffused. Note that the word S_0 will always have a zero difference independent of the number of **R** transforms we apply in this attack as this word is always truncated to insert the message word. Hence, our differential characteristic from $\mathbf{G2}_{11}$ gets diffused into the 29-word state after the application of full **G** and 6 rounds of **R** that include last 2 rounds to process length-padding words. This clearly demonstrates weak state diffusion in **F**-256.

Remark 2. The differential characteristic of our free-start distinguisher can also be viewed as an improved variant of the characteristic considered by the designers in the analysis of the PRF mode of **F**-256 [6, §12.4.2]. The internal state they have considered at the start of the **G** transform for the partial collisions differ in the same words as ours and in an additional word of S_0. As shown in Section 6.1, with negligible computational work we can have S_0 with zero difference.

Remark 3. We note that one can also extend the distinguisher of Aumasson and Phan on the $\tau(\mathbf{G}(S))$ transform to produce a free-start distinguisher for **F**-256. Our analysis shows that this extension can produce pairs of input states in the distinguishing attack that differ alike in 9 words (of which 5 have 0 difference) in round -2 of **R**, in 4 words (of which 3 have 0 difference) in round -3 of **R**, in 2 words with 0 difference in round -4 of **R** and the states get fully diffused in round -5 and in this round the distinguisher does not hold anymore. The rounds -1 and 0 of **R** are meant for the length-padding words that denote the binary representation of messages of size at most 3 words such that the distinguisher can hold. However, this extension would never produce equal length message words because the word S_0 at the **G** always has a difference activating the $\overline{\mathbf{SMIX}}$-es of round 0 of **R**. Hence, the message word in Round 0 will always have a difference meaning that the message words produced by the distinguisher can be at most of size 2^{27} words (i.e 2^{32} bits). As we just noted, the distinguisher cannot hold for more than 5 rounds of **R** that include the last two length-padded rounds. Hence, for this extension to apply to a maximum of 5 rounds of **R**, we need to use freedom in the round $\mathbf{G2}_{12}$ such that we control the unequal length padding of word 0 for pairs of states. This takes about 2^{34} work leading to more than 66-bit input difference at the start of the **G** transform. The desired length padding in Round -1 can be obtained with an approach similar to our analysis.

7 On free-start collisions for F-256

Recall from §6.3 that free-start distinguishers for the length-padded **F**-256 also produce free-start collisions. In this section, we show that we do not need to develop efficient differential characteristics for the **G** transform to show free-start collisions for the length-padded **F**-256 and these collisions for **F**-256 can be obtained in constant time “trivially’ even with the inclusion of length-padding. First we discuss some generic methods of doing free-start collisions for **F**-256.

[5] Here we do not consider the word S_0 as it is always be replaced by a message word in a round of **R**.

7.1 Generic free-start collisions

Free-start collisions without length padding. It is well known that the structure of any instance of Fugue allows trivial free-start collisions in constant time and negligible memory even if **G** is ideal. This is possible by inverting a pair of distinct *final states* that produce the same digests through **G** and at least one round of the **R** transform. However, to find generic pseudo collisions for Fugue with length padding some computational work is required as shown below. We focus on Fugue-256 (with length-padding) and recall that the **R** transforms are numbered in the order $0, -1, -2, \ldots$. These were not discussed before.

Free-start collisions with equal length words. In this attack, on average 2^{64} distinct *final states* (that produce the same digest) are inverted through **G** and 2 rounds of **R** for each message in order to obtain the desired length-padded words in the last 2 rounds of **R**. Hence, using brute force, it takes on average 2^{64} hashing operations to find free-start collisions for equal length message words. This attack produces colliding messages of desired size.

Free-start collisions with unequal length words. In this attack, to obtain each colliding message, we invert about 2^{32} *final states* such that we obtain $0x00000000$ in the round -1 of the **R** transform. We repeat this to obtain another message in the colliding pair. Both these messages will likely have distinct 32-bit length padding words in Round 0 of **R**. Now we invert the states of this colliding pair to generate unequal length messages of size at most 2^{32}. So total complexity of this attack is at most 2^{34}. This attack does not lead to desired length of message words in the pseudo collisions.

7.2 Free-start collisions for the length-padded F-256 in constant time.

The following algorithm shows a free-start collision attack for **F**-256 in constant time where we can get colliding messages of desired size. Let $0, -1, -2, \ldots$ be the **R** transforms where length padding words are processed in the rounds -1 and 0 and **G** transform follows 0^{th} **R** transform.

1. Consider the 29-word state of the -1^{th} **R** transform and this state has words $S_{1\sim 29}$. Fill in these words with arbitrary values. Invert 29-word state together with some word S_0 through the round -2 and obtain a message word m^{-2}.
2. Now the word S_0 of the 30-word state at this round can be varied such that $S_0 \oplus S_{10}$ is always fixed to the value of S_{10} in the 29-word state chosen in the previous step. By using this freedom in the word S_0 and inverting the 29-word state chosen in round -1 of the above step, we can obtain a message $m'^{-2} \neq m^{-2}$ in round -2 of **R**.
3. Now we have found a pair of distinct message words (m'^{-2}, m^{-2}) that collide on the 29-word state at the -1^{th} **R** transform. Both of them will have the same length encoded words in the rounds -1 and 0 and hence, would give a collision after the $\tau(\mathbf{G})$ transform.
4. Invert the state at round -2 to the desired number of rounds and obtain colliding message words and states in every round. Finally, length encode the message words in the last two rounds of **R** producing an internal state collision and hence the free-start collision for the length-padded **F**-256.

8 Concluding remarks

Fugue is probably the least studied hash function among the second round of SHA-3 candidates. Fugue is based on a very new mode of operation, and the security properties required on its building blocks were not clearly defined. Hence, improved analysis of this design to develop a deeper understanding of it is important. We made this attempt in this paper for the 256-bit instance of Fugue called Fugue-256, denoted **F**-256.

We improved the previous integral distinguisher on the *final transformation* **G** from 5.5 rounds to 16.5 rounds showing weak diffusion in this transform. Next, we improved the meet-in-the-middle preimage attack of the designers and reduced its complexity from 2^{480} time and memory to 2^{416}. Next, we developed an improved version of the differential characteristic used by the designers in the PRF mode analysis of **F**-256 to develop free-start structural distinguishers and free-start collisions for **F**-256. This analysis complements the designers' analysis of the chosen key/Initial Value (IV) model of **F**-256 [5]. Finally, we show efficient "trivial" free-start collisions of different type for **F**-256 *without* the necessity of the distinguishers for the *final transformation* **G**.

To conclude, in this paper we have developed a further understanding of the design of **F**-256. Although our new analytical results do not compromise the security claims of the designers for finding collisions and (second) preimages in these designs as well as its indifferentiability analysis [7], they show several interesting properties of these designs such as weak diffusion in **G** transform through integrals and efficient distinguishers, weak diffusion for the composition of few rounds of **R** and **G** transforms in **F**-256 through free-start distinguishers and free-start collisions and the ability to control message and state words as in the meet-in-the-middle preimage attack on **F**-256.

Acknowledgments: We would like to thank Charanjit Jutla and Shai Halevi for their many valuable discussions and comments on the analysis presented in this paper. We also thank Raphael Phan, Jean-Philippe Aumasson, Søren Thomsen and Christian Rechberger for comments and discussions.

Praveen Gauravaram is supported by the Danish Council for Independent Research: Technology and Production Sciences (FTP) 09-066486/FTP.

The work in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

Lei Wei is supported under the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03. An earlier version of this paper was submitted to the NIST Hash function email listing in December 2010. This is a revised version of that paper with some additional analysis and corrections to the previous version.

References

1. J.-P. Aumasson and R. C.-W. Phan. On the Cryptanalysis of the Hash Function Fugue: Partitioning and Inside-Out Distinguishers. To appear in IPL Journal,2011.
2. J.-P. Aumasson and R. C.-W. Phan. Distinguisher for Full Final Round of Fugue-256. Presented at second NIST SHA-3 conference, 2010.
3. J. Daemen and V. Rijmen. *The design of Rijndael: AES — the Advanced Encryption Standard*. Springer, 2002.
4. ECRYPT II. The SHA-3 Zoo. Available at `http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo`.
5. S. Halevi, W. E. Hall, and C. S. Jutla. Analysis of Fugue-256. Comment to NIST's hash function forum on April 4, 2010.
6. S. Halevi, W. E. Hall, and C. S. Jutla. The Hash Function Fugue. Submission to NIST (updated), 2009.
7. S. Halevi, W. E. Hall, C. S. Jutla, and A. Roy. Weak Ideal Functionalities for Designing Random Oracles with Applications to Fugue. Submission to NIST's function forum in 2010.
8. D. Khovratovich. Cryptanalysis of hash functions with structures. In M. J. J. Jr., V. Rijmen, and R. Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2009.
9. L. Knudsen and D. Wagner. Integral cryptanalysis. In *FSE*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.
10. L. R. Knudsen, C. Rechberger, and S. S. Thomsen. The Grindahl Hash Functions. In A. Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 39–57. Springer, 2007.
11. NIST. Announcing the Development of New Hash Algorithms for the Revision of Federal Information Processing Standard (FIPS) 180-2, Secure Hash Standard, January 2007. This notice by NIST is available at `http://www.csrc.nist.gov/pki/HashWorkshop/timeline.html` with the Docket No: 061213336-6336-01. (Accessed on 2/11/2010).
12. NIST. Second Round Candidates. Official notification from NIST, 2009. Available at `http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html` (Accessed on 17/02/2011).
13. NIST. Status Report on the Second Round of the SHA-3 Cryptographic Hash Algorithm Competition, February 2011. This Interagency Report 7764 of NIST is available at `http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/documents/Round2_Report_NISTIR_7764.pdf` (Accessed on 17/02/2011).
14. M. S. Turan and E. Uyan. Near-Collisions for the Reduced Round Versions of Some Second Round SHA-3 Compression Functions Using Hill Climbing. In G. Gong and K. C. Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010*, volume 6498 of *Lecture Notes in Computer Science*, pages 131–143. Springer, 2010.

A Differential path and examples of free-start structural distinguisher for F-256

The differential path which demonstrates free-start structural distinguisher for Fugue-256 is shown in Figure 1. This path shows the differences in the state words (represented as square cells) at the start and the end of every full round of the **G** and **R** transforms. The cells in Magenta color represent δ_1, those in Blue color represent δ_2 and those in Red color represent random differences. Cells in Light Green color are zero difference words chosen at the start of the *Forwards* phase of the attack with their actual values fixed from the round $\mathbf{G2}_{11}$ and plain cells in the round $\mathbf{G2}_{11}$ are zero differences whose *actual values* are varied in the attack. The cells in Forest Green and Yellow at the 0^{th} **R** transform denote the words S_0 and S_{10} on which we exert the control. The cells in Black color are those whose actual values and hence the difference remains fixed when the words $S_{8\sim14}$ and $S_{22\sim29}$ of the round $\mathbf{G2}_{11}$ are varied for the attack.

Table 1 shows two pairs of free-start initial states that produce the same digest difference. The message words in the Round -2 for the first pair of states are $0x452e0fed$ and $0xb69c87e7$ respectively and their hash values H_1 and H_1^* and digest difference $H_1 \oplus H_1^*$ are noted in Table2. The message words in the Round -2 for the second pair of states are $0x3e38f1d5$ and $0x3b3e1591$ respectively and their hash values H_2 and H_2^* and digest difference $H_2 \oplus H_2^*$ are noted in Table2.

B Integrals for the 16.5 rounds of G transform of F-256

In Table 3, we have shown the integrals for the 16.5 rounds of the **G** transform of **F**-256 whose analysis was presented in §4.

Table 1. Two pairs of *free-start initial states* at Round -2 that produce the same digest difference under **F**-256. The underlined differences (thirteen words) in each pair are the same, thus illustrating a symmetrical structure at the input. The message word m^{-2} for the first pair of states are $452e0fed$ and $b69c87e7$ respectively and the message word m^{-2} for the second pair of states are $3e38f1d5$ and $3b3e1591$ respectively. Digests and their difference for these two pairs of free-start initial states are noted in Table 6.

(First pair)i	0	1	2	3	4	5	6	7
S_i	00000000	60384cec	c88579f6	ef3384a7	1122cf88	699f49c9	061dd5c7	d0ec2932
S_i^*	00000000	1b4b4b3b	c88579ff	bf715080	1122cf88	699f49c9	061dd5ce	80aefd15
δS_i	00000000	7b7307d7	00000009	5042d427	00000000	00000000	00000009	5042d427
i	8	9	10	11	12	13	14	15
S_i	490b3310	e9515c99	0f318596	5d5cf9cb	8b3ed0d8	f24df5be	0e882d53	a04d1f33
S_i^*	bab9bb1a	3f0e6e44	3060b6a1	952343d8	8b3ed0d8	f24df5b7	5ecaf974	a04d1f33
δS_i	f3b2880a	d65f32dd	3f513337	c87fba13	00000000	00000009	5042d427	00000000
i	16	17	18	19	20	21	22	23
S_i	55c282ac	7dac1e94	8128362d	9bfb7773	e027a681	26b47a29	31432623	1d8fd2d0
S_i^*	55c282ac	7dac1e9d	c653bcab	8fcd7448	c8a9b641	4e4ee6a6	6c9e69b9	b5612464
δS_i	00000000	00000009	477b8a86	1436033b	288e10c0	68fa9c8f	5ddd4f9a	a8eef6b4
i	24	25	26	27	28	29		
S_i	e6beba51	f632b2d6	6a9e272e	48be89d7	415cc7c3	d0913503		
S_i^*	9dcdbd86	4d01224b	4e845d7d	c0be7791	6ba488f9	651ff74e		
δS_i	7b7307d7	bb33909d	241a7a53	8800fe46	2af84f3a	b58ec24d		

(Second pair)i	0	1	2	3	4	5	6	7
S_i	00000000	dfc0c8dc	9f377939	79a8eedf	a1494d9a	7c7ec219	57ba13ef	419425ff
S_i^*	00000000	a384d860	9f377930	29ea3af8	a1494d9a	7c7ec219	57ba13e6	11d6f1d8
δS_i	00000000	7c4410bc	00000009	5042d427	00000000	00000000	00000009	5042d427
i	8	9	10	11	12	13	14	15
S_i	92a4950a	ef401399	24a6b3da	885f29bd	d2ad82df	d4a1c996	61e5f82c	f5df2e9c
S_i^*	97a2714e	df695c53	dc179321	e45d6bef	d2ad82df	d4a1c99f	31a72c0b	f5df2e9c
δS_i	0506e444	30294fca	f8b120fb	6c024252	00000000	00000009	5042d427	00000000
i	16	17	18	19	20	21	22	23
S_i	6c43db27	a3d7bba2	ff80e3c3	e6e9d43f	7545d720	9ef2661d	f366637c	2643cd41
S_i^*	6c43db27	a3d7bbab	b85bdda6	8ee38551	b8356608	f0253cc4	7cfb5757	5a907f38
δS_i	00000000	00000009	47db3e65	680a516e	cd70b128	6ed75ad9	8f9d342b	7cd3b279
i	24	25	26	27	28	29		
S_i	b2a6c9a6	23253798	8cdb1796	cd9ed766	d6336037	251d81be		
S_i^*	cee2d91a	3acd494c	42729eb7	3e270858	8160fad6	bb1c154c		
δS_i	7c4410bc	19e87ed4	cea98921	f3b9df3e	57539ae1	9e0194f2		

Table 2. Digests and their difference for the pair of free-start initial states presented in Table 1. The digests for the first pair are (H_1, H_1^*) and those for the second pair are (H_2, H_2^*). Note that $H_1 = H_2$ and $H_1^* = H_2^*$ showing free-start collisions.

H_1	H_1^*	δH_1	H_2	H_2^*	δH_2
c7d79278	1110cc99	d6c75ee1	c7d79278	1110cc99	d6c75ee1
5bf7c4c7	0c8414e5	5773d022	5bf7c4c7	0c8414e5	5773d022
89887088	a2001d55	2b886ddd	89887088	a2001d55	2b886ddd
d766450d	e3e3bdcf	3485f8c2	d766450d	e3e3bdcf	3485f8c2
9832fbba	8aeada67	12d821dd	9832fbba	8aeada67	12d821dd
1af7391d	df11a14c	c5e69851	1af7391d	df11a14c	c5e69851
b81725a5	ab74f777	1363d2d2	b81725a5	ab74f777	1363d2d2
c073bb41	4d788f18	8d0b3459	c073bb41	4d788f18	8d0b3459

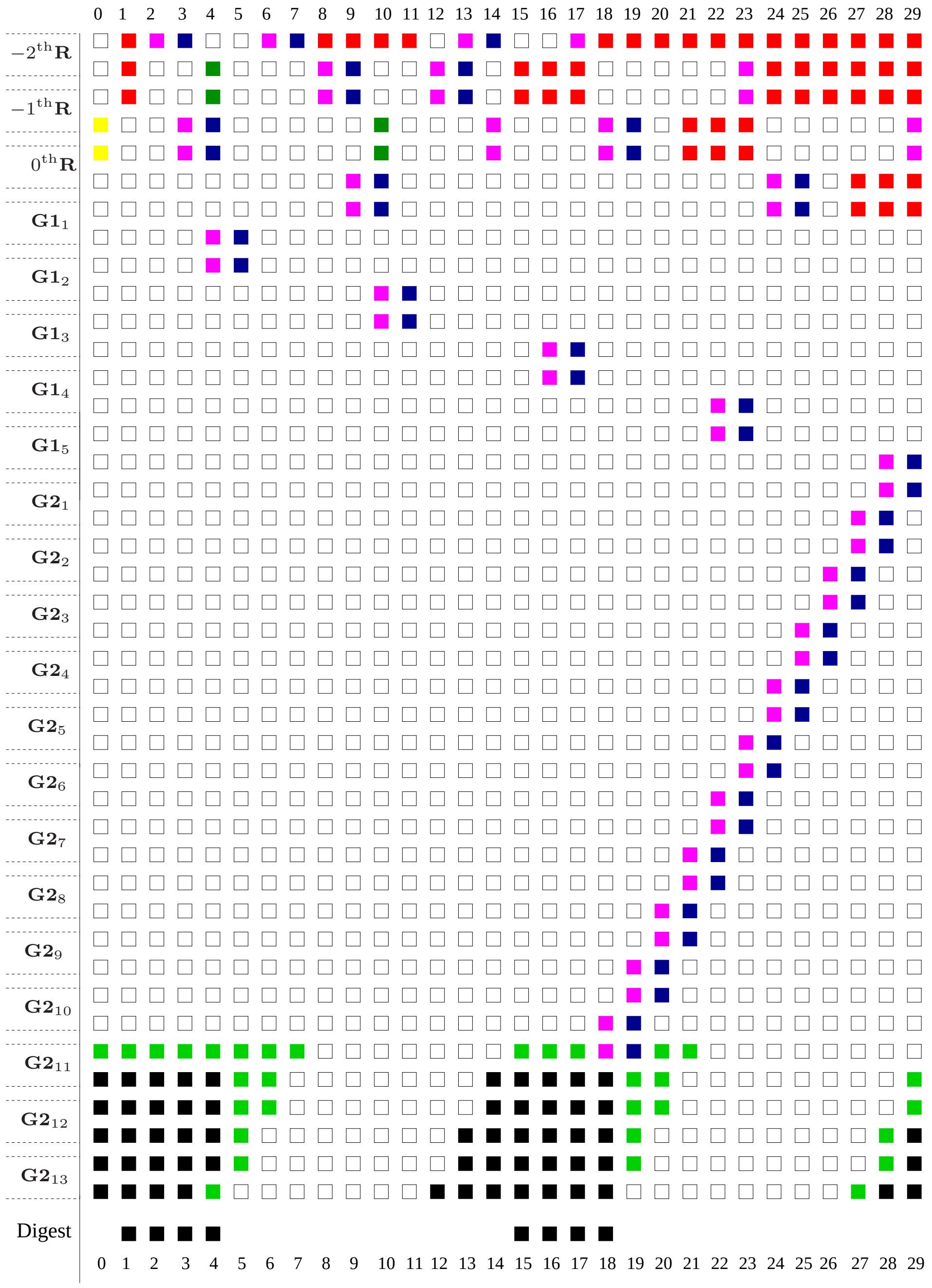

Fig. 1. Differential path of the free-start distinguisher on **F**-256. At Round -2 of **R**, the two pairs of *pseudo initial states* differ alike in the words S_0,$S_{2\sim 7}$ and $S_{12\sim 17}$.

Table 3. The Integral propagation over 16.5 rounds of **F**-256 **G**-function. In this table "." , "," and "!" denote $\mathcal{C}$, $\mathcal{A}$ and $\mathcal{S}$ respectively and blank cells are those cells that we do not consider.

	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}	S_{21}	S_{22}	S_{23}	S_{24}	S_{25}	S_{26}	S_{27}	S_{28}	S_{29}
G_1^1	.	.	.	.	.,..	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
$G_1^{1.5}$	.	.	.	.	.	.	.	.,..	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
G_1^2	.	.	.	.	.	.	.	.	.	.	.,..	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
$G_1^{2.5}$	.	.	.	.	.	.	.	.	.	.	.	.	.	.,..	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
G_1^3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.,..	.	.	.	.	.	.	.	.	.	.	.	.	.
$G_1^{3.5}$	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.,..	.	.	.	.	.	.	.	.	.	.
G_1^4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.,..	.	.	.	.	.	.	.
$G_1^{4.5}$	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.,..	.	.	.	.
G_1^5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.,..	.
$G_1^{5.5}$	.,..	,...	...,	..,.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
G_2^1	,!,!	,!,!	,,,,	,!,!	,...	...,	..,.	.	.	.	.	.	.	.	.	,...	...,	..,.	.	.	.	.	.	.	.	.	.	.	.	.
$G_2^{1.5}$	??!?	,?,?	,???	.?!?	.	.	.	.	.	.	.	.	.	.	.	,!,!	,!,!	,,,,	,!,!	!!,!	...,	..,.	.	.	.	.	.	.	.	.
G_2^2															??!?															
$G_2^{2.5}$																														??!?
G_2^3														??!?																
$G_2^{3.5}$																													??!?	
G_2^4													??!?																	
$G_2^{4.5}$																												??!?		
G_2^5												??!?																		
$G_2^{5.5}$																											??!?			
G_2^6											??!?																			
$G_2^{6.5}$																										??!?				
G_2^7										??!?																				
$G_2^{7.5}$																									??!?					
G_2^8									??!?																					
$G_2^{8.5}$																								??!?						
G_2^9								??!?																						
$G_2^{9.5}$																							??!?							
G_2^{10}							??!?																							
$G_2^{10.5}$																						??!?								
G_2^{11}						??!?																								
$G_2^{11.5}$																					??!?									
G_2^{12}					??!?																									
$G_2^{12.5}$																				??!?										