Rebound Attack on Reduced-Round Versions of JH

Vincent Rijmen[1,*], Deniz Toz[1,*] and Kerem Varıcı[1,*]

[1] Katholieke Universiteit Leuven
Department of Electronical Engineering ESAT SDC-COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{vincent.rijmen,deniz.toz,kerem.varici}@esat.kuleuven.be

Abstract. JH, designed by Wu, is one of the 14 second round candidates in the NIST Hash Competition. This paper presents the first analysis results of JH by using rebound attack. We first investigate a variant of the JH hash function family for $d = 4$ and describe how the attack works. Then, we apply the attack for $d = 8$, which is the version submitted to the competition. As a result, we obtain a semi-free-start collision for 16 rounds (out of 35.5) of JH for all hash sizes with $2^{179.24}$ compression function calls. We then extend our attack to 19 rounds and present a 1008-bit semi-free-start near-collision on the JH compression function with $2^{156.77}$ compression function calls, $2^{152.28}$ memory access and $2^{143.70}$-bytes of memory.

1 Introduction

Recent years witnessed the continuous works on analysis of hash functions which reveal that most of them are not as secure as claimed. Wang et al. presented collisions on the MD family [1, 2, 3] using an attack technique on hash functions which is based on differential cryptanalysis. This idea was further developed and used in the analysis of other famous and widely used hash functions SHA-1 and SHA-2 [4, 5, 6]. In response, the National Institute of Standards and Technology (NIST) announced a public competition for designing a new hash function which will be chosen as the hash function standard: Secure Hash Algorithm 3 (SHA-3) [7].

JH [8] is the submission of Hongjun Wu to the NIST Hash Competition and it is one of the 14 second round candidates. According to the designer, the hash algorithm is very simple and efficient in both software and hardware. JH supports four different hash sizes (224, 256, 384 and 1024-bit), and is among the fastest contestants.

*This work was sponsored by the Research Fund K.U.Leuven, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT Programme under Contract ICT-2007-216676 (ECRYPT II). The information in this paper is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Table 1. Summary of results for JH (CFC - Compression Function Call)

	#Rounds	Time Complexity	Memory Complexity	Collision Type	Sec.
Hash Func.	16	$2^{178.24}$ CFC	$2^{101.12}$ byte	semi-free-start collision	§3.2
Comp. Func.	19	$2^{156.77}$ CFC	$2^{143.70}$ byte	semi-free-start near-collision (1008 bits)	§4.2
Comp. Func.	22	$2^{156.56}$ CFC	$2^{143.70}$ byte	semi-free-start near-collision (768 bits)	§4.2

The rebound attack [9], a new technique for cryptanalysis of hash function, is introduced by Mendel et al. in FSE 2009. It is applicable to both block cipher based and permutation based hash constructions. Later, it is improved by Mendel et al. [10] and Matusiewicz et al. [11]. In this work, we bring all these ideas together to analyze the JH hash function. First, we implement the rebound attack to the small scale variant of JH by choosing $d = 4$. Then, we adapt the method to the submitted version of JH ($d = 8$). In order to improve the complexity of the attack, we use three inbound phases rather than one, which provides us to use freedom more efficiently. The results can be seen in Tab. 1.

This paper is organized as follows: In Sec. 2, we give a brief description of the JH hash function, its properties and an overview of the rebound attack. In Sec. 3, we first describe the main idea of our attack on small scale version of JH and then give the results on submitted version of JH. In Sec. 4, we follow the same outline for the improved version of the rebound attack. Finally, we conclude this paper and summarize our results in Sec. 5.

2 Preliminaries

2.1 Notation

Throughout this paper, we will use the following notation:

word	4-bit
$m_{i,j}$	the j^{th} word of the i^{th} round value
d	dimension of a block of bits
	i.e. a d-dimensional block of bits consists of 2^d words
JH-X	the member of the family whose message digest is X bits
$\|\|$	concatenation operation
$\times$	cross-product: an operation on two arrays that result in another array whose elements are obtained by combining each element in one array with every element in the second one

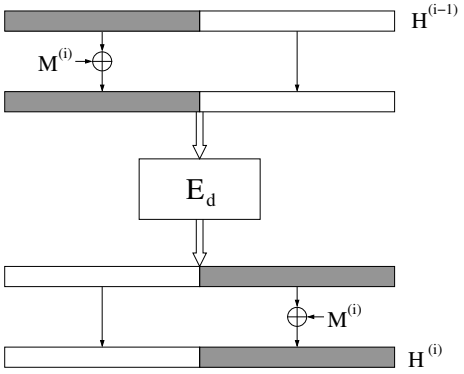

Fig. 1. The compression function F_d

2.2 The JH Hash Function

The hash function JH is an iterative hash function that accepts message blocks of 512 bits and produces a hash value of 224, 256, 384 and 512 bits. The message is padded to be a multiple of 512 bits. The bit '1' is appended to the end of the message, followed by $384 - 1 + (-l \bmod 512)+$ zero bits. Finally, a 128-bit block is appended which is the length of the message, l, represented in big endian form. Note that this scheme guarantees that at least 512 additional bits are padded. In each iteration, the compression function F_d, given in Fig.1, is used to update the 2^{d+2} bits as follows:

$$H_i = F_d(H_{i-1}, M_i)$$

where H_{i-1} is the previous chaining value and M_i is the current message block. The compression function F_d is defined as follows:

$$F_d(H_{i-1}, M_i) = E_d(H_{i-1} \oplus M_i || 0^{2^{d+1}}) \oplus 0^{2^{d+1}} || M_i.$$

Here, E_d is a permutation and is composed of an initial grouping of bits followed by $5(d-1)$ rounds, plus an additional S-Box layer and a final degrouping of bits. The grouping operation arranges bits in a way that the input to each S-Box has two bits from the message and two bits from the chaining value. In each round, the input is divided into 2^d words and then each word passes through an S-Box. JH uses two 4-bit-to-4-bit S-Boxes ($S0$ and $S1$) and every round constant bit selects which S-Boxes are used. Then two consecutive words pass through the linear transformation L, which is based on a $[4, 2, 3]$ Maximum Distance Separable (MDS) code over $GF(2^4)$. Finally all words are permuted by the permutation P_d. After the degrouping operation each bit returns to its original position. The round function for $d = 4$ is shown in Fig. 2 and $d = 8$ is the submitted version. For a more detailed information we refer to the specification of JH [8].

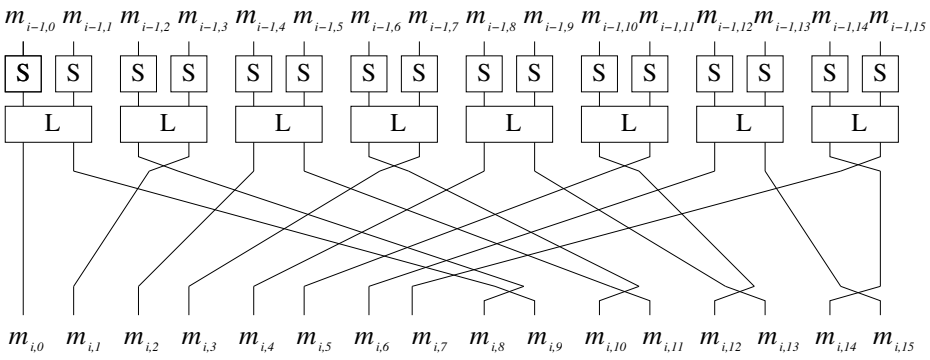

Fig. 2. Round Function for $d = 4$

The initial hash value H_0 is set depending on the message digest size. The first two bytes of H_1 are set as the message digest size, and the rest bytes of H_1 are set as 0. Finally, the message digest is generated by truncating H_N where N is the number of blocks in the padded message., i.e, the last X bits of H_N are given as the message digest of JH-X for $X = 224, 256, 384, 512$.

2.3 Properties of the Linear Transformation L

Since the linear transformation L implements a $(4, 2, 3)$ MDS matrix, any difference in one of the words of the input (output) will result in a difference in two words of the output (input). If one tries all possible 2^{16} pairs, the number of pairs satisfying each condition ($2 \rightarrow 1$ or $1 \rightarrow 2$) is 3840, which gives a probability of $3840/65536 \approx 2^{-4.09}$. Note that, if the words are arranged in a way that they will be both active this probability increases to $3840/57600 \approx 2^{-3.91}$. For the latter case, if both words remain active ($2 \rightarrow 2$), the probability is $49920/57600 \approx 2^{-0.21}$.

2.4 Observations on the Compression Function

The grouping of bits at the beginning of the compression function assures that the input of every first layer S-Box is xor-ed with two message bits. Similarly, the output of each S-Box is xor-ed with two message bits. Therefore, for a random non-zero 4-bit difference, the probability that this difference is related to a message is $3/15 \approx 2^{-2.32}$.

The bit-slice implementation of F_d uses $d - 1$ different round function descriptions. The main difference between these round functions is the permutation function. In each round permutation, the odd bits are swapped by $2^r \bmod (d-1)$ where r is the round number. Therefore, for the same input passing through multiple rounds, the output is identical to the output of the original round function

for the $\alpha \cdot (d-1)$-th round where α is any integer. Three rounds of the bit-sliced representation can be seen in Fig. 3 between rounds 1 and 4.

2.5 The Rebound Attack

The rebound attack is introduced by Mendel et al. [9]. The two main steps of the attack are called inbound phase and outbound phase. In the inbound phase, the freedom is used to connect the middle rounds by using the match-in-the-middle technique and in the outbound phase connected truncated differentials are calculated in both forward and backward direction.

This attack has been first used for the cryptanalysis of reduced versions of Whirlpool and Grøstl, and then extended to obtain distinguishers for the full Whirlpool compression function [12]. Later, linearized match-in-the-middle and start-from-the-middle techniques are introduced by Mendel et al. [10] to improve the rebound attack. Moreover, a sparse truncated differential path and state is recently used in the attack on LANE by Matusiewicz et al. [11] rather than an all active state in the matching part of the attack.

In our work, we first apply the start-from-the-middle technique [10] with an all active state, then we improve our results by using three inbound phases with partially active states rather than one all active one. This allows us to decrease the complexity requirements of the attack.

3 The Start-From-The-Middle Attack on JH

We use the available freedom in the middle by a start-from-the-middle-technique. We begin by guessing the middle values and then proceed forwards and backwards using the filtering conditions to reduce the number of active S-Boxes in each round.

In this section, we describe the steps of our attack on JH in detail. We will first describe the attack on smaller version of JH, i.e., $d = 4$ in detail, and then give the algorithm and analysis for $d = 8$.

3.1 Attack on 8 Rounds of JH for $d = 4$

Attack Procedure: The inbound phase of the attack described in this section is composed of 8 rounds, and the number of active S-Boxes in each round is:

$$1 \leftarrow 2 \leftarrow 4 \leftarrow 8 \leftarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

The bit-slice implementation allows us to analyze the algorithm easily. The truncated differential path is given in Fig. 3. The attack can be summarized as follows:

- **Step 1:** Try all possible 2^{16} values for the middle values $m_{4,j}||m_{4,j+1}$ and $m'_{4,j}||m'_{4,j+1}$ in Round 4 for each of the four sets (shown with colors and different shapes in Fig. 3), and keep only the ones that satisfy the desired pattern ($2 \rightarrow 4 \rightarrow 2$). Therefore, the expected number of remaining pairs is $2^{16} \cdot 2^{16}/[(2^{4.09})^2 \cdot (2^{3.91})^2] = 2^{16}$ for each set.

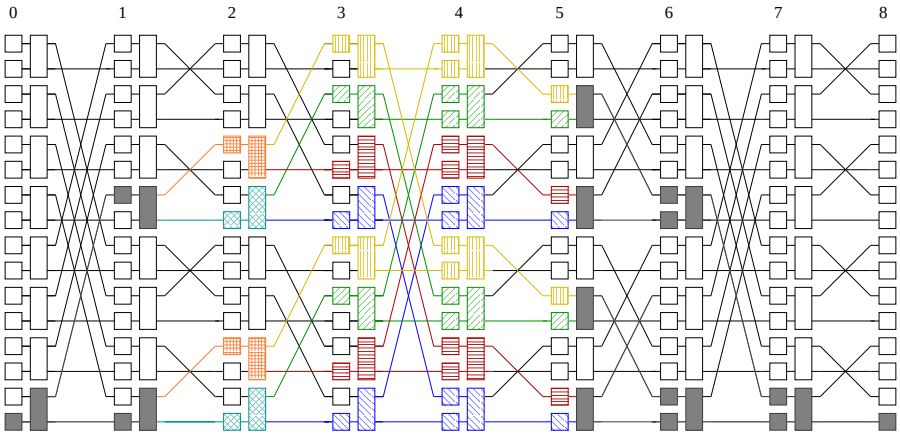

Fig. 3. Inbound Phase of JH for $d = 4$ (bit-slice implementation)

- **Step 2:** Compute the cross-product of the sets: ▨ × ▨ and ▤ × ▥, check if the differences satisfy $2 \rightarrow 1$ when they pass the inverse L transform, L^{-1}. For the pairs that satisfy the filtering condition, store only the values in the active words and the middle values for each of the 2 sets. After this step, the number of pairs in each set is approximately $2^{16} \cdot 2^{16}/(2^{3.91})^2 = 2^{24.18}$.
- **Step 3:** Compute the cross-product of the sets: ▩ × ▦, check whether the remaining 10 filtering conditions (marked with ▬) are satisfied or not. This control can be done by calculating $L \circ S$ or $S^{-1} \circ L^{-1}$ for only the active words and does not require the use of the round function entirely, hence it is very efficient. The total number of remaining pairs that pass the inbound phase is $2^{24.18} \cdot 2^{24.18}/(2^{3.91})^{10} = 2^{9.26}$.

Note that, due to the symmetry, the actual number of remaining pairs is $2^{8.26}$ and the duplication can be avoided in the earlier steps of the algorithm, but for simplicity it is described like this in the paper. The attack algorithm only stores the middle values for the pairs that follows the desired differential path and the values in the n-th round can be computed by calling the round (or inverse round) function.

The active S-Boxes in the input and output to the compression function must satisfy the desired property, so out of $2^{8.26}$ pairs only $2^{8.26} \cdot (2^{-2.32})^2 = 2^{4.32}$ of them remain. In order to obtain a collision, the difference in both S-Boxes also need to match, which happens with a probability of $1/3$. Therefore for $2^{4.32} \cdot 1/3 \simeq 2^{2.74}$ pairs we obtain a semi-free-start collision for the hash function.

Complexity of the Attack: The inbound phase is the part of the attack where most of the calculations is done. Let $U = L \circ S$ and $U^{-1} = S^{-1} \circ L^{-1}$. Then,

a round function consists of 8 U-functions, similarly an inverse round functions has 8 U^{-1}-functions.

For each of the four sets in Step 1 of the algorithm, we try all possible 2^{16} pairs and apply the filtering condition, Although we have 2 U-functions in forward direction and 2 U^{-1}-functions in backward direction, we only check one condition if the previous one is satisfied, so the total number of calls, n_1 is:

$$n_1 = 2^{32} + 2^{32}/2^{4.09} + 2^{32}/(2^{4.09})^2 + 2^{32}/[(2^{4.09})^2 \cdot 2^{3.91}] = 2^{32.09}$$

which is approximately 2^{291} round functions. However this step can be done by 2^{32} table look-ups if precomputation is used.

For each of the two sets in Step 2 of the algorithm, since L is a linear transformation (so does L^{-1}), it is sufficient to check whether the differences satisfy the desired property, i.e.$(L^{-1}(\Delta c, \Delta d) = (\Delta a, 0)$ or $(0, \Delta b))$. The total number of calls in this step is:

$$n_2 = (2^{16})^2 \cdot (1 + 2^{-3.91}) = 2^{32.09}$$

In the final step of the inbound phase, 3 of the 10 conditions to be checked are again linear transformations and the remaining 7 require the use of U-function. The complexity of the attack is dominated by this step and the total number of operations performed is:

$$n_{3,bck} = (2^{24.18})^2 \cdot [(1 + 2^{-3.91} + (2^{-3.91})^2] \simeq 2^{48.46}$$

$$n_{3,fwd} = (2^{24.18})^2/(2^{-3.91})^3 \cdot \sum_{i=0}^{6} (2^{-3.91})^i \simeq 2^{36.72}$$

where fwd and bck denote the forward and backward direction respectively. In the outbound phase, for each of the $2^{8.26}$ remaining pairs, starting from the middle values, we call the round and inverse round functions the obtain the input and output values.

Results: The above algorithm has been implemented in C for $d = 4$ and we observed that the results are compatible with the precomputed values. Thus, we obtain semi-free-start collision for 8-rounds JH-16. An example is given in Tab. 5.

3.2 The Attack on 16 Rounds of JH with $d = 8$

In this section, we first present an outline for the start-from-the middle attack on reduced round version of JH for all hash sizes, and then give the calculations for the complexity analysis of the attack.

[1] $2^{32.09} \cdot 1/8 = 2^{29.09}$

Table 2. The overview of inbound phase

Step	Size (bits)	Sets	Filtering Conditions	Pairs Remain	Time Complexity	Direction
0	8	128	1	$2^{11.75}$	$2^{15.84}$	*fwd*
1	16	64	1	$2^{19.59}$	$2^{23.50}$	*bck*
2	32	32	4	$2^{23.54}$	$2^{39.18}$	*fwd*
3	64	16	4	$2^{31.44}$	$2^{47.08}$	*fwd*
4	128	8	4	$2^{47.24}$	$2^{62.88}$	*fwd*
5	256	4	8	$2^{63.20}$	$2^{94.48}$	*fwd*
6	512	2	8	$2^{95.12}$	$2^{124.40}$	*fwd*
7	1024	1	46	$2^{10.38}$	$2^{190.24}$	*fwd + bck*

Attack Procedure: For the compression function E_8, the attack is composed of 16 rounds and the number of active S-Boxes is:

$$1 \leftarrow 2 \leftarrow 4 \leftarrow 8 \leftarrow 16 \leftarrow 32 \leftarrow 64 \leftarrow 128 \leftarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

The algorithm is similar to the one of E_4. We again start from the middle and then propagate outwards by computing the cross-product of the sets and using the filtering conditions. However, instead of trying all 2^{16} possible pairs, we start with $2^{7.92}$ values for each middle value. The number of sets, the bit length of the middle values (size) of each set, and the number of filtering conditions followed by the number of pairs in each set are given in Tab. 2. Similarly, we only store the values in the active bytes for the outermost rounds and the middle round for each set, i.e., no other intermediate value is stored.

Complexity of the Attack: The time complexity of the attack for $d = 8$ is calculated in a manner similar to that of $d = 4$. Instead of giving all equations explicitly, we summarize the results in terms of function calls and their direction for each step in Tab. 2. The time complexity of the given attacks is $2^{190.24}$ U-function calls (equivalent to $2^{190.24} \cdot 2^{-7} \cdot (1/16) = 2^{179.24}$ 16-round compression function calls).

Results: We may lose up to half of the remaining pairs due to symmetry. In addition, similar to the case for d=4, the active S-Boxes in the input and output to the compression function should correspond only to the message bits and then match each other in order to obtain a collision. Therefore, out of $2^{10.38}$ pairs only $2^{10.38} \cdot 1/2 \cdot (2^{-2.32})^2 \cdot 1/3 \simeq 2^{3.15}$ of them remain.

Suppose that we intend to attack a block M_i where $(i < N)$. Since we obtain a zero difference in the chaining value that guarantees that the outputs of the compression function will be the same provided that the both messages have the same length. As mentioned earlier the same compression function is used for all hash sizes, and the message digest is generated by truncating H_N where N is

the number of blocks in the padded message. Therefore, we have semi-free-start collision for all hash sizes of 16-round JH.

4 The Rebound Attack on JH Compression Function

The attack on the hash function given in Sec. 3.2 can easily be converted to an attack on 19 rounds of the compression function for the pairs that satisfy the inbound phase by using the following differential trails in the outbound phases:

$$2 \leftarrow 1 \leftarrow \text{ Inbound Phase } \rightarrow 1 \rightarrow 2 \rightarrow 4$$

The complexity of the attack remains same (i.e., $2^{190.24}$ U-function calls) and we obtain a semi-free-start near-collision for 1008 bits. In this section, we improve these results by using three inbound phases. Once again, we first describe the steps of our attack for $d = 4$ in detail, and then give the algorithm and complexity analysis for $d = 8$.

4.1 The Improved Rebound Attack on JH with d= 4

The inbound phase of the attack described in this section is composed of 8 rounds, using the following trail:

$$2 \leftarrow 4 \rightarrow 2 \leftarrow 4 \leftarrow 8 \rightarrow 4 \leftarrow 8 \rightarrow 4 \rightarrow 2 \tag{1}$$

It is perhaps interesting to make here some observations on the number of active S-boxes in the trail. Similar to the AES, the linear diffusion layer of JH imposes a lower bound on the number of active S-boxes: if $d \geq 2$, then there are at least $3^2 = 9$ active S-boxes in every sequence of 4 rounds. The conjectured bound on the number of active S-boxes over $2d + 1$ rounds [8], as well as the trail (1), demonstrate that the *higher dimension* of the JH diffusion layer allows for relatively long and *narrow trails*.

We decompose the inbound phase into a sequence of three smaller inbound phases, each of which are 3, 2 and 3 rounds respectively. The number of active SBoxes for each of the steps in each round is:

$$\begin{aligned} &2 \leftarrow 4 \rightarrow 2 \\ &2 \leftarrow 4 \leftarrow 8 \rightarrow 4 \\ &4 \leftarrow 8 \rightarrow 4 \rightarrow 2 \end{aligned}$$

We use the bit-sliced representation to analyze the algorithm. We first calculate the results of the first and the third inbound phases, and then match them with the second inbound phase. The truncated differential path is given in Fig. 4. The attack can be summarized as follows:

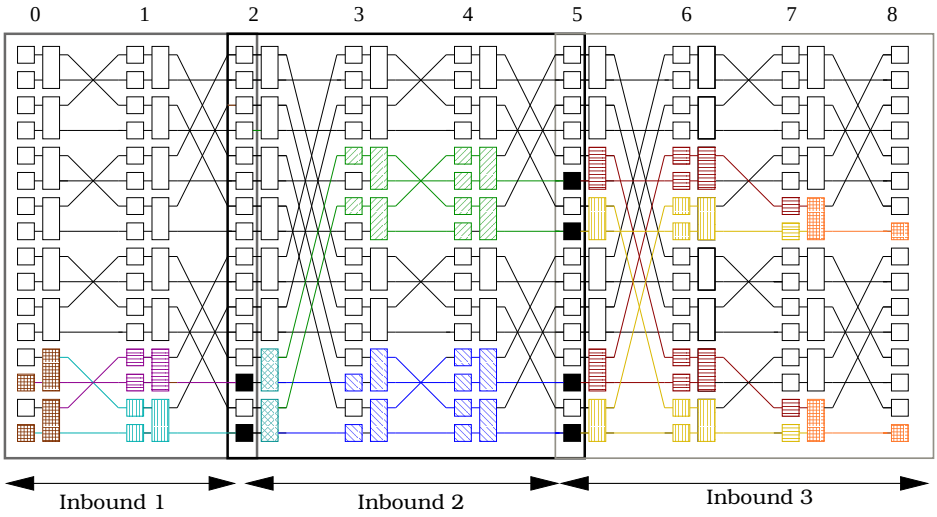

Fig. 4. Inbound Phase of JH ($d = 4$)

First Inbound Phase:

- Try all possible 2^8 values for each of the middle values $m_{1,j}||m_{1,j+1}$ and $m'_{1,j}||m'_{1,j+1}$ in Round 1 for each of the two sets, and keep only the ones that satisfy the desired pattern. Therefore, the expected number of remaining pairs is $2^8 \cdot 2^8/2^{4.09} = 2^{11.91}$ for each set.
- Compute the cross-product of the two sets, check if the differences satisfy $2 \rightarrow 1$ when they pass L^{-1}. For the pairs that satisfy the filtering condition, store only the values in the active words and the middle values. After this step, the number of pairs is approximately $2^{11.91} \cdot 2^{11.91}/(2^{3.91})^2 = 2^{16}$.
- Check whether the remaining pairs satisfy the desired input difference, and store these values in list L_1. Therefore, the size of L_1 is $2^{16} \cdot (2^{-2.32})^2 = 2^{11.36}$

Second Inbound Phase:

- Try all possible 2^8 values for each of the middle values $m_{4,j}||m_{4,j+1}$ and $m'_{4,j}||m'_{4,j+1}$ in Round 4 for each of the four sets, and keep only the ones that satisfy the desired pattern. The expected number of remaining pairs is again $2^8 \cdot 2^8/2^{4.09} = 2^{11.91}$ for each set.
- Compute the cross-product of the sets having the same pattern, check if the differences satisfy $2 \rightarrow 1$ when they pass L^{-1}. For the pairs that satisfy the filtering condition, store the values for each of the 2 sets. The expected number of pairs in each set is approximately $(2^{11.91})^2/(2^{3.91})^2 = 2^{16}$.

- Compute the cross-product of the sets, ▧ × ▨, and check if the differences satisfy the filtering condition when they pass L^{-1}. The expected number of remaining pairs that pass the second inbound phase is $(2^{16})^2/(2^{-3.91})^2 = 2^{24.18}$.

Third Inbound Phase:

- Try all possible 2^8 values for each of the middle values $m_{6,j}||m_{6,j+1}$ and $m'_{6,j}||m'_{6,j+1}$ in Round 6 for each of the four sets, and keep only the ones that satisfy the desired pattern. The expected number of remaining pairs is again $2^{11.91}$ for each set.
- Compute the cross-product of the sets having the same pattern, check if the differences satisfy $2 \rightarrow 1$ when they pass the inverse L transform. For the pairs that satisfy the filtering condition, store the values for each of the 2 sets. The expected number of pairs in each set is approximately $(2^{11.91})^2/(2^{3.91})^2 = 2^{16}$.
- Compute the cross-product of the two sets, ▤ × ▥, to check the final filtering conditions in round 7. The expected number of pairs that pass the third phase is $(2^{16})^2/(2^{-3.91})^2 = 2^{24.18}$. Store these values in list L_3.

Merging Inbound Phases: The three previous inbound phases overlap in the 2 and 4 active words (denoted with black) in rounds 2 and 5 respectively. Since we have to match these active words in both values, we get a condition on $16 + 32 = 48$ bits in total. These conditions are checked as soon as we have a remaining pair for the second inbound phase, by using the lists L_1 and L_3. As a result, we expect to find $(2^{11.36} \cdot 2^{24.18} \cdot 2^{-16}) \cdot 2^{24.18} \cdot 2^{-32} \cdot (2^{-2}) \simeq 2^{9.72}$ solutions for the inbound phase. The last 1/4 factor in the calculation follows from symmetry.

Outbound Phase: For the pairs that satisfy the inbound phase, we expect to see the following differential trail in the outbound phase:

$$2 \leftarrow \text{ Inbound Phase } \rightarrow 2 \rightarrow 4 \rightarrow 2$$

Therefore, for the compression function E_4, we have the 10-round differential path shown in Fig. 5. Note that, there are two filtering conditions in the last round of the outbound phase. Thus, out of $2^{9.72}$ solutions, only $2^{9.72} \cdot (2^{-3.91})^2 \approx 2^{1.9}$ pass to the last round. After the degrouping operation, the message is xor-ed to the rightmost 32-bits of the output and for the compression function of JH for $d = 4$ we have a near-collision for 52 bits.

Data Complexity: The time complexity of the attack is determined by the first and third inbound phases which is about $2^{32.09}$ each, hence the total time complexity is $2^{32.09} + 2^{32.09} = 2^{33.09}$ U-function calls, equivalent to $2^{33.09} \cdot 2^{-3} \cdot (1/10) = 2^{26.77}$ compression function calls. The memory complexity is also determined by the third inbound phase which is $2^{24.18}$.

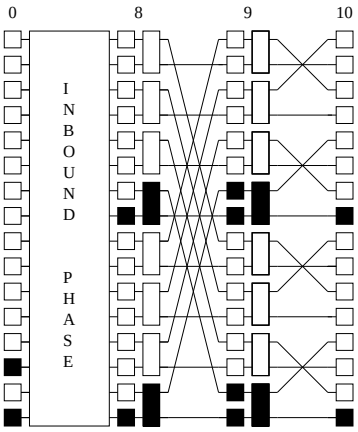

Fig. 5. Outbound Phase of JH ($d = 4$)

Results: We obtain a 52-bit free-start near-collision for 10 rounds of the JH compression function. The results are still not better than theoretic bounds for JH with d = 4, but it helps us to implement the attack and expand it for the submitted version of JH.

4.2 The Improved Rebound Attack on JH with $d = 8$

In this section, the attack in Sec. 4.1 on JH with $d = 4$ is extended to JH with $d = 8$ using more rounds (hence the larger number of steps and the increased complexity) for each of the inbound and outbound phases. The attack is applicable to 19 rounds of the compression function. We first explain the attack in detail and then give the calculations for the complexity analysis.

Inbound Phase: For the compression function E_8, the inbound phase of the attack is 16 rounds and is composed of two parts. In the first part, we apply the start-from-the-middle-technique three times for rounds $0-3$, $3-10$ and $10-16$. In the second part, we connect the resulting active bytes (hence the corresponding state values) by a match-in-the-middle step. The number of active S-Boxes in each of the sets is:

$$
\begin{aligned}
&2 \leftarrow 4 \leftarrow 8 \rightarrow 4\\
&4 \leftarrow 8 \leftarrow 16 \leftarrow 32 \leftarrow 64 \leftarrow 128 \rightarrow 64 \rightarrow 32\\
&32 \leftarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2
\end{aligned}
$$

For a detailed sketch of the inbound phase one can refer to Fig. 7 (Appendix). The algorithm for each set is similar to the one of E_4. We again start from the middle and then propagate outwards by computing the cross-product of the sets

Table 3. The overview of inbound phases

Step	Size	Sets	Filtering Conditions	Pairs Remain	Complexity Backwards	Complexity Forwards
0	8	4	1	$2^{11.91}$	–	2^{16}
1	16	2	2	2^{16}	$2^{23.91}$	–
2	32	1	2	$2^{24.18}$	$2^{32.09}$	–
3	32	1	2^a	$2^{19.54}$		
0	8	64	1	$2^{11.91}$	–	2^{16}
1	16	32	2	2^{16}	$2^{23.91}$	–
2	32	16	2	$2^{24.18}$	–	$2^{32.09}$
3	64	8	4	$2^{32.72}$	$2^{48.46}$	–
4	128	4	4	$2^{49.80}$	$2^{65.54}$	–
5	256	2	4	$2^{83.96}$	$2^{99.70}$	–
6	512	1	4	$2^{152.28}$	$2^{168.02}$	–
0	8	32	1	$2^{11.91}$	–	2^{16}
1	16	16	2	2^{16}	$2^{23.91}$	–
2	32	8	2	$2^{24.18}$	–	$2^{32.09}$
3	64	4	2	$2^{40.54}$	–	$2^{48.45}$
4	128	2	2	$2^{73.26}$	–	$2^{81.17}$
5	256	2	2	$2^{138.70}$	–	$2^{146.61}$

[a] Check whether the pairs satisfy the desired input difference

and using the filtering conditions. For each list, we try all possible 2^{16} pairs in Step 0. The number of sets, the bit length of the middle values (size) of each list, and the number of filtering conditions followed by the number of pairs in each set are given in Tab. 3.

Merging Inbound Phases: Connecting these three lists is again performed as follows. Whenever a pair is obtained from set 2, we check whether it exits in L_1 or not. If it does, another check is done for L_3. Since we have $2^{19.54}$ and $2^{138.7}$ elements in lists 1 and 3 respectively, $2^{152.28}$ pairs passing the second inbound phase, and 32-bit and 256-bit conditions for the matches. The total expected number of remaining pairs is $(2^{19.54} \cdot 2^{152.28} \cdot 2^{-32}) \cdot 2^{138.70} \cdot 2^{-256} \cdot (2^{-2}) = 2^{20.52}$.

We obtained more pairs than usual due to the additional filtering conditions in the outbound phase, in order to obtain a near-collision (which will be explained in the following part) for 19 rounds of the compression function.

Outbound Phase: The outbound of the attack is composed of 3 rounds in the forward direction. For the pairs that satisfy the inbound phase, we expect to see the following differential trail in the outbound phase:

$$\text{Inbound Phase} \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 4$$

Table 4. Complexity of the generic attack for near-collisions

#Rounds	# bits Near Collision	Generic Attack Complexity	Our Results
19	1008	$2^{454.21}$	$2^{156.77}$
20	992	$2^{411.18}$	$2^{156.70}$
21	960	$2^{341.45}$	$2^{156.63}$
22	896	$2^{236.06}$	$2^{156.56}$
23	768	$2^{99.18}$	$2^{156.50}$

Note that in the last step of the outbound phase we have four filtering conditions. We had $2^{20.52}$ remaining pairs from the inbound phase, thus, we expect $2^{20.52} \cdot (2^{3.91})^4 = 2^{4.88}$ pairs to satisfy the above path. A detailed schema of this trail is shown in Fig. 7 (Appendix).

The final step of the compression function is xor-ing the message bits after the degrouping operation to the output of the compression function. We have 4 active words in the output and 4-bit difference in the message, two of which collide in bit positions 512 and 768. Thus, it is possible to cancel them with a probability of 2^{-2} and the number of pairs reduce to $2^{2.88}$. To sum up, we have a difference in $(4 \cdot 4 - 2) + 2 = 16$ bits in total.

Complexity of the Attack: For the inbound phase, the complexity of the attack for $d = 8$ is calculated in a similar manner to that of $d = 4$. The results for preparing the lists are summarized for each step in Tab. 3. The time complexity of the attack is dominated by the second set, L_2, which is about $2^{168.02}$ U-function calls (equivalent to $2^{168.02} \cdot 2^{-7} \cdot (1/19) = 2^{156.77}$ 19-round compression function calls). The memory requirements are determined by the largest list, which is L_3 with a size of $2^{138.70}$ 256-bit data.

Results: Note that, in this attack, the complexity requirements are reduced significantly compared to the initial idea that uses only one inbound phase. For 19 rounds of the JH compression function, we obtain a semi-free-start near-collision for 1008 bits. We can simply increase the number of rounds by proceeding forwards in the outbound phase. Our attack still works in this case with the same complexity (U-function calls). The number of bits for near-collision and the generic attack complexities are given in Tab. 4. As a result, our attack is better than generic attacks up to 22 rounds.

5 Conclusion

In this paper, we presented the first cryptanalysis results of JH by using rebound attack techniques. We first explained our attack on 8 rounds of JH ($d = 4$) in detail and then showed how this attack can be used in order to attack 16 rounds

of JH hash function. We further improved our findings by using three inbound phases instead of one and obtained a 1008-bit semi-free-start near-collision for 19 rounds of the JH compression function. The required memory for the attack is $2^{143.70}$ bytes of data. The time complexity is reduced to $2^{156.77}$ compression function calls, and it requires $2^{152.28}$ memory accesses. We also presented semi-free-start near-collision results for 20-22 rounds of the JH compression function with the same memory requirement and time complexity. Our findings are summarized in Tab. 1.

References

1. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. [13] 1–18
2. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. [13] 19–35
3. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In Shoup, V., ed.: CRYPTO. Volume 3621 of LNCS., Springer (2005) 17–36
4. Cannière, C.D., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In Lai, X., Chen, K., eds.: ASIACRYPT. Volume 4284 of LNCS., Springer (2006) 1–20
5. Cannière, C.D., Mendel, F., Rechberger, C.: Collisions for 70-step sha-1: On the full cost of collision search. In Adams, C.M., Miri, A., Wiener, M.J., eds.: Selected Areas in Cryptography. Volume 4876 of Lecture Notes in Computer Science., Springer (2007) 56–73
6. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A.K., Molnar, D., Osvik, D.A., de Weger, B.: Short chosen-prefix collisions for md5 and the creation of a rogue ca certificate. In Halevi, S., ed.: CRYPTO. Volume 5677 of Lecture Notes in Computer Science., Springer (2009) 55–69
7. NIST: Cryptographic Hash Competition `http://www.nist.gov/hash-competition`.
8. Wu, H.: The Hash Function JH. Submission to NIST (2008) `http://icsd.i2r.a-star.edu.sg/staff/hongjun/jh/jh_round2.pdf`.
9. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Dunkelman, O., ed.: FSE. Volume 5665 of Lecture Notes in Computer Science., Springer (2009) 260–276
10. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved cryptanalysis of the reduced grøstl compression function, echo permutation and aes block cipher. In Jr., M.J.J., Rijmen, V., Safavi-Naini, R., eds.: Selected Areas in Cryptography. Volume 5867 of Lecture Notes in Computer Science., Springer (2009) 16–35
11. Matusiewicz, K., Naya-Plasencia, M., Nikolic, I., Sasaki, Y., Schläffer, M.: Rebound attack on the full lane compression function. [14] 106–125
12. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: Results on the full whirlpool compression function. [14] 126–143
13. Cramer, R., ed.: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of LNCS., Springer (2005)
14. Matsui, M., ed.: Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. In Matsui, M., ed.: ASIACRYPT. Volume 5912 of Lecture Notes in Computer Science., Springer (2009)

A Sample Data

We performed several experiments for different S-Box setups, but the implementation results shown below are obtained by using the same S-Box ($S0$) in each round for simplicity.

Table 5. Example for rebound attack with one inbound phase ($d = 4$)

	Bit-Slice Results			Reference Results		
	P_1	P_2	Difference	P_1	P_2	Difference
0	6ddf8804acec67ef	addf8804acec67ef	c000000000000000	6dacdfec886704ef	adacdfec886704ef	c000000000000000
1	53d4792d4304231c	03d4792d4104231c	5000000002000000	53d4792d4304231c	03d4792d4104231c	5000000002000000
2	c7b01dd79c2227e3	e7ba1dd7cc2927e3	200a0000500b0000	c71d9c27b0d722e3	e71dcc27bad729e3	200050000a000b00
3	55c5a1f5a7f8a0bc	3595acfe6708a9b2	60500d0bc0f0090e	55a7c5f8a1a0f5bc	35679508aca9feb2	60c050f00d090b0e
4	2b3ead8b7712b433	a9f04e08299bd357	82cee3835e896764	2b3ead8b7712b433	a9f04e08299bd357	82cee3835e896764
5	bd2e0491b4824e41	d62ef09101827441	6b00f400b5003a00	bd04b44e2e918241	d6f001742e918241	6bf4b53a00000000
6	7c47294041d9f1c6	46472940b0d9f1c6	3a000000f1000000	7c4147d929f140c6	46b047d929f140c6	3af1000000000000
7	89100fddca5632e0	a5100fddca5632e0	2c00000000000000	89100fddca5632e0	a5100fddca5632e0	2c00000000000000
8	29184f8fc8fc9af1	19184f8fc8fc9af1	3000000000000000	294fc89a188ffcf1	194fc89a188ffcf1	3000000000000000

B Differential characteristic for 16 rounds of JH Hash Function

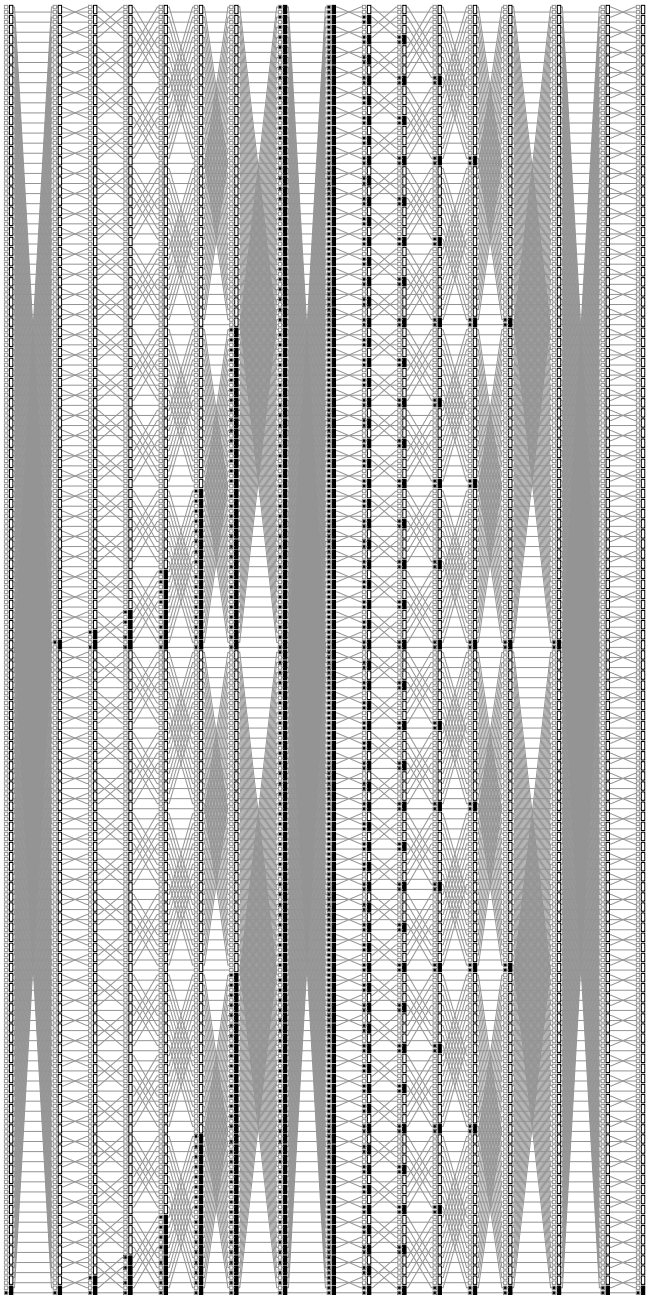

Fig. 6. Differential characteristic for 16 rounds of JH Hash Function (bit-slice representation)

C Inbound and Outbound Phases for Compression Function E_8

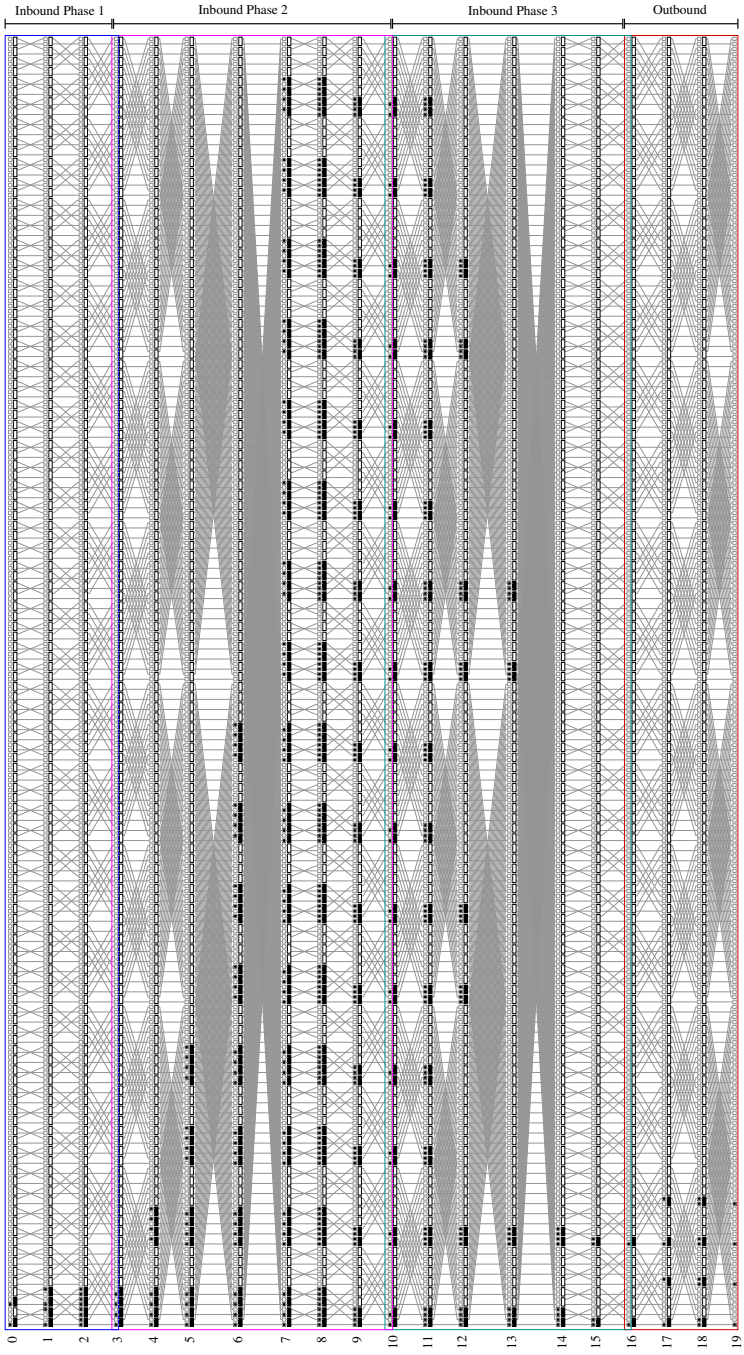

Fig. 7. Inbound and Outbound Phases of JH compression function (bit-slice representation)