Cryptanalysis of the LSH and SHA-V Hash Functions

Yonglin Hao, Hongbo Yu

Department of Computer Science and Technology, Tsinghua Universtiy, Beijing 100084, China
haoyl14@mails.tsinghua.edu.cn
yuhongbo@mail.tsinghua.edu.cn

Abstract. In this paper, we study the security of two hash function families LSH and SHA-V. We find that the wide-pipe MD structural LSH hash functions do not apply the traditional feeding forward operation. This structural weakness enables us to launch free-start collision and pseudo-preimage attacks on full-round LSH hash functions with negligible complexities.
In order to evaluate the quality of the LSH round functions, we launch 14-round boomerang attacks on LSH-512 and LSH-256 hash functions with complexities 2^{308} and 2^{242} respectively. We verify the correctness of our boomerang attacks by giving practical 11-round boomerang quartets. These boomerang results indicate that the round functions of LSH are well designed.
Based on our analysis, we stress that the adoption of the feeding forward operation should be essential to the LSH hash functions despite of their well designed round functions.
The PMD structural SHA-V parallelizes two SHA-1-like streams and each stream processes independent 512-bit message blocks. This structure enable us to utilize the divide-and-conquer strategy to find preimages and collisions. Our preimage attack can be applied to full-round SHA-V with time & memory complexities $O(2^{80})$. Our trivial collision attacks also requires $O(2^{80})$ complexities but, utilizing existing results on SHA-1, we can find a SHA-V collision with a time complexity $O(2^{61})$ and a negligible memory complexity. These results indicate that there are weaknesses in both the structure and the round function of SHA-V.

Keywords: Hash Function, Boomerang Attack, LSH, SHA-V, MD Structure, Feeding Forward

1 Introduction

Cryptographic hash functions (simply referred as hash functions) are playing a significant role in the modern cryptology. An ideal hash function meet three criterions namely: preimage resistance, 2nd preimage resistance and collision resistance. In 2005, Wang et al. successfully launched collision attacks on widely used hash functions MD5 [1] and SHA-1 [2] which forced NIST to proposed the transition from SHA-1 to SHA-2. However, doubts on the security have been continuously raised that SHA-2 may also be vulnerable to such attacks due to similar design approach to the attacked hash functions. To cope with this situation, in the year 2007, NIST launched the SHA-3 competition [3] to develop a new hash standard. This competition largely stimulated the cryptanalysis technique on hash functions. After years' analysis, five proposals entered the final round of SHA-3 and the one named Keccak became the new SHA-3 standard in 2012 [4].

The end of the SHA-3 competition does not end the proposal of new hash function designs. Although it has been selected as the new SHA-3 standard because of its distinct design and better hardware efficiency, Keccak shows relatively low software performance compared to other SHA-3 candidates. When much more and bigger data needs to be hashed in the era of smart devices, implementing cryptographic algorithm at the hardware level will be the main trend without doubt. However, the hardware implementation will not be able to have the competitive edge in price to the software one without large quantity production. Furthermore, the software implementation has many advantages in terms of management, flexibility, portability, ease of use/upgrade, etc. [5]. Therefore, a hash function with good software performance would be more marketable when considering the present and the near future. The LSH [6], a new hash function family proposed by Kim et al. at ICISC 2014, is a design in accordance with such circumstances and considerations. As a software oriented hash function, LSH has two versions namely LSH-256 and LSH-512, suitable for 32- and 64-bit word processors respectively. In the original introduction of LSH [6], the designers have given thorough evaluations to the security of LSH against different attacking models. But we find their analysis is still insufficient. In this paper, we revisit the secure margins of the LSH hash functions in two ways. Firstly, we give some trivial attacks based on some structural weaknesses of LSH. Then, we evaluate the strength of the round function by launching boomerang attacks on LSH hash functions.

Another hash function family we are going to study is SHA-V [7]. Derived from SHA-1, SHA-V can be regarded two parallelized SHA-1 streams. It updates 320-bit chaining variables and has 7 versions denoted as SHA-V-$(128 + 32k)$ where $k \in [0, 6]$ and $(128 + 32k)$ represents the bitwise output length. SHA-V has not received too much attention ever since its proposal. But recently at Eurocrypt 2015, Leurent et al. present a generic preimage attack on the XOR combiner of two independent hash functions [8] using complicated structures such as multi-collisions [9]. In [8], the authors specifically mentioned that their method can be applied to directly to SHA-V-160 with complexity $\tilde{O}(2^{133.3})$. We show that SHA-V is even weaker so that we can break all of its 7 versions.

Related Works. One of the main method used in this paper is the boomerang attack. It was introduced by Wagner in 1999 [10] as a tool for the cryptanalysis of block ciphers. During the past few years, the idea of the boomerang attack has been applied to many hash functions and turned out to be quite fruitful. Biryukov et al. [11] and Lamberger et al. [12] independently applied the boomerang attack to BLAKE-32 and SHA-256. The SHA-256 result was later improved by Biryukov et al. in [13]. Ever after, we saw the boomerang results on many hash functions such as SIMD-512 [14], HAVAL [15], RIPEMD-128/160 [16], HAS-160 [15], Skein-256/512 [17,18], SM3 [19,20], BLAKE-256/512 [21,22] and BLAKE2 [22]. The boomerang attack has become a common tool for analyzing various hash functions.

As to the boomerang attack on LSH, the designers claim "*we can construct 16-step and 17-step boomerang distinguishers [53] for LSH- 256 and LSH-512, respectively, by combining short differential characteristics*"[6]. According to the authors, the 16- and 17-round distinguishers requires complexities 2^{468} and 2^{772} respectively, which exceeds the generic bounds 2^{256} for LSH-256 and 2^{512} for LSH-512. Furthermore, according to our analysis, the direct concatenation of two short differential characteristics results in many contradictive conditions which makes the 16- and 17-round characteristics unavailable. Therefore, it is a left-open question that how many rounds can an available boomerang distinguisher reach for the LSH hash functions within the generic bounds. We are to answer this question in this paper.

Our Contributions. We find that, as a (wide-pipe) MD structural hash function, LSH has omitted the traditional feeding forward operation in its compression functions. This structural weakness enables us to launch free-start collision and pseudo-preimage attacks on full-round LSH hash functions with negligible complexities. In order to estimate the LSH round functions, we construct available differential characteristics and launch boomerang attacks on 14-round LSH-512 and LSH-256 hash functions with complexities 2^{308} and 2^{242} respectively. We verify the the correctness of our attacks by giving practical boomerang quartets for 11-round LSH hash functions. To the best of our knowledge, these are the first practically verifiable boomerang results on the LSH hash functions.

For SHA-V, we find that the two SHA-1-like streams of SHA-V are processing independent 512-bit message blocks. Therefore, we launch our preimage attacks on full-round SHA-V-$(128 + 32k)$ for all versions $k \in [0, 6]$. The time and memory complexities of our attacks are $O(2^{80})$. We also propose collision attacks on on full-round SHA-V-$(128 + 32k)$ for $k \in [2, 6]$ with time & memory complexities $O(2^{80})$. Utilizing existing SHA-1 results, we can eliminate the memory complexity and lower the time complexity to $O(2^{61})$. This improved collision attack can then be applied to all SHA-V versions.

Organization of the Paper. In Section 2, we briefly introduce LSH and SHA-V, and provide the overview of the boomerang attack. We reveal the structural weakness of LSH hash functions by presenting some trivial attacks on the full-round versions in Section 3. Section 4 describes our Type I and III boomerang attacks on round-reduced LSH-512 and LSH-256. We present our preimage and collision attacks on SHA-V in Section 5. Finally, we conclude our paper in Section 6.

2 Preliminary

We briefly introduce LSH and SHA-V hash function families in the first two parts of this section. In the third part, we introduce the three types of the boomerang attack and review the procedure of the widely-used differential-based boomerang attack on hash functions.

2.1 Brief Introduction of the LSH Hash Functions

Some notations have to be introduced first:

$\leftarrow$ variable assignment;
$+$ modular 2^{32} or 2^{64} addition (according to the word length);
$-$ modular 2^{32} or 2^{64} subtraction (according to the word length);
$\oplus$ bitwise exclusive or;
$\lll n$ cyclic shift n bits towards the most significant bit;
$\ggg n$ cyclic shift n bits towards the least significant bit;
$\wedge$ bitwise AND operation for words;
$\mathcal{W}^t$ the set of all t-word arrays ($t \geq 1$). In this paper, let $\mathcal{W}$ denote $\mathcal{W}^1$;
$LSB_n(\cdot)$ getting the least significant n bits of a bit string.

The hash function family LSH consists of n-bit hash functions based on w-bit word, $\{$LSH-$8w$-$n : w = 32 \text{ or } 64, 1 \leq n \leq 8w\}$. LSH-$8w$-$n$ has the wide-pipe MD structure with one-zeros padding. The message hashing process of LSH-$8w$-n consists of the following three stages:

Initialization: The given bit string message m is padded and cut into $t = \lceil \frac{|m|+1}{32w} \rceil$ 32-word message blocks, denoted as $M^{(0)}, \cdots, M^{(t-1)}$. The 16-word chaining variable array $CV^{(0)}$ is initialized to the constant initialization vector IV of LSH-$8w$-n.

Compression: Updating of chaining variables by iteration of a compression function $CF : \mathcal{W}^{16} \times \mathcal{W}^{32} \rightarrow \mathcal{W}^{16}$ with message blocks $\{M^{(i)}\}_{i=0}^{t-1}$, which means computing $CV^{(i)}$ ($i \in [1, t]$) as

$$CV^{(i)} = CF(CV^{(i-1)}, M^{(i-1)}) \tag{1}$$

and acquiring the final chaining variable $CV^{(t)}$. We will describe CF in detail later in this section.

Finalization: The finalization function FIN_n return n-bit hash value h from the final chaining variable $CV^{(t)} = (CV_0^{(t)}, \cdots, CV_{15}^{(t)})$. FIN_n first compute a 8-word array hash H as

$$H = (H_0, \cdots, H_7) = (CV_0^{(t)} \oplus CV_8^{(t)}, \cdots, CV_7^{(t)} \oplus CV_{15}^{(t)}) \tag{2}$$

and then return the n-bit hash value h as

$$h = LSB_n(H_0 \| \cdots \| H_7).$$

Specifically, when $n = 8w$, we have $h = H_0 \| \cdots \| H_7$ providing the highest secure margin.

For the simplicity of interpretation, we only consider the two most secure versions with $n = 8w$, which are LSH-256-256 and LSH-512-512 for 32- and 64-bit word respectively. Since the final output string h is only a concatenation of the words in the hash array H, we only consider H as the output without specific declarations. In the remainder of this paper, we denote LSH-256-256 simply by LSH-256 and LSH-512-512 by LSH-512.

The compression function of LSH, denoted by CF can be regarded as

$$CF : \mathcal{W}^{16} \times \mathcal{W}^{32} \rightarrow \mathcal{W}^{16}$$

where $\mathcal{W}$ refers to the 64-bit words for LSH-512 or 32-bit words for LSH-256. The following four functions are used in a compression function:

1. $ME : \mathcal{W}^{32} \rightarrow \mathcal{W}^{16(N_s+1)}$ (Message expansion function),
2. $MA : \mathcal{W}^{16} \times \mathcal{W}^{16} \rightarrow \mathcal{W}^{16}$ (Message addition function),
3. $MX_r : \mathcal{W}^{16} \rightarrow \mathcal{W}^{16}$ (Mix function of the r-th round),
4. $WP : \mathcal{W}^{16} \rightarrow \mathcal{W}^{16}$ (Word permutation function),

where $N_s = 28$ for LSH-512 and $N_s = 26$ for LSH-256, and $r \in [0, N_s]$. The round function of round r, denoted by $F_r : \mathcal{W}^{16} \times \mathcal{W}^{16} \rightarrow \mathcal{W}^{16}$, can be defined as

$$F_r = WP \circ MX_r \circ MA.$$

We detail the four functions as follows.

Message Expansion Function. Firstly, the ME function a 32-word message block, denoted as $M = (M_0, \cdots, M_{31})$ to $N_s + 1$ 16-word word arrays denoted as $W^0, \cdots, W^{N_s}$ where

$$W^r = (W_0^r, \cdots, W_{15}^r), \quad r \in [0, N_s].$$

The first two word arrays W^0 and W^1 are initialized by M as

$$W^0 = (M_0, \cdots, M_{15}), \quad W^1 = (M_{16}, \cdots, M_{31}).$$

For $r \in [2, N_s]$, the word array W^r is generated by

$$W_l^r \leftarrow W_l^{r-1} + W_{\tau(l)}^{r-2}, \quad l \in [0, 15].$$

where τ is the permutation over $\mathbb{Z}_{16}$ defined in Table 1.

Table 1. The definition of permutations τ and δ.

l	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\tau(l)$	3	2	0	1	7	4	5	6	11	10	8	9	15	12	13	14
$\delta(l)$	6	4	5	7	12	15	14	13	2	0	1	3	8	11	10	9

Message Addition Function. For two 16-word arrays $X = (X_0, \cdots, X_{15})$ and $Y = (Y_0, \cdots, Y_{15})$,, the message addition function $MA : \mathcal{W}^{16} \times \mathcal{W}^{16} \to \mathcal{W}^{16}$ is defined as follow:

$$MA(X, Y) := (X_0 \oplus Y_0, \cdots, X_{15} \oplus Y_{15}). \tag{3}$$

Mix Function. The r-th mix function $MX_r : \mathcal{W}^{16} \to \mathcal{W}^{16}$ updates the 16-word array $T = (T_0, \cdots, T_{15})$ by mixing every two word pair (T_l, T_{l+8}) for $l \in [0, 7]$. For $r \in [0, N_s - 1]$, the mix function $MX_{r,l}$ proceeds .

$$(T_l, T_{l+8}) \leftarrow MX_{r,l}(T_l, T_{l+8}), \tag{4}$$

where $MX_{r,l}$ is a two-word mix function. Let X and Y be two words. The two-word mix function $MX_{r,l}(X, Y) = (X', Y')$ is defined by (5), where α_r, β_r and γ_l are defined in Table 2. As to the round constants SC_l^r, we refer the readers to [6] for detailed definitions.

$$\begin{aligned} aX &\leftarrow X + Y, \\ bX &\leftarrow (X \oplus SC_l^r)^{\lll \alpha_r}, \\ aY &\leftarrow bX + Y, \\ bY &\leftarrow Y^{\lll \beta_r} \\ X' &\leftarrow bX + bY \\ Y' &\leftarrow bY^{\lll \gamma_l}. \end{aligned} \tag{5}$$

Table 2. The definition of rotation amounts α_r, β_r and γ_l.

	r	α_r	β_r	γ_0	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7
LSH-512	even	23	59	0	16	32	48	8	24	40	56
	odd	7	3								
LSH-256	even	29	1	0	8	16	24	24	16	8	0
	odd	5	17								

Word-Permutation Function. Let $X = (X_0, \cdots, X_{15})$ be an 16-word array. The word-permutation function $WP : \mathcal{W}^{16} \to \mathcal{W}^{16}$ is defined by

$$WP(X) := (X_{\delta(0)}, \cdots, X_{\delta(15)}), \tag{6}$$

where δ is the permutation over $\mathbb{Z}_{16}$ defined by Table 1.

Given the 16-word variable $CV^{(i)}$ ($i \in [0, t-1]$) defined in (1), a 16-word variable T^0 is initialized as

$$T^0 = (T_0^0, \cdots, T_{15}^0) = (CV_0^{(i)}, \cdots, CV_{15}^{(i)}). \tag{7}$$

After the assignment of T^0, F_r computes a new 16-word array $T^{r+1} = F_r(T^r, M_r)$ for $r \in [0, N_s]$. For the convenience of interpretation, we denote the intermediate value after the $r-th$ MA function as U^r, MX_r as V^r, then we have

$$F_r : T^r \xrightarrow{MA(\cdot, W^r)} U^r \xrightarrow{MX_r} V^r \xrightarrow{WP} T^{r+1}.$$

The process of $CV^{(i+1)} = CF(CV^{(i)}, M^{(i)})$ ($i \in [0, t-1]$) can be described as Algorithm 1.

```
Algorithm 1: The Compression Function $CF$ of LSH
  Input: A 16-word chaining variable $CV^{(i)}$ and a 32-word message block $M^{(i)}$ ($i \in [0, t-1]$).
  Output: A 16-word chaining variable $CV^{(i+1)}$
  Initialize $T^0 \leftarrow CV^{(0)}$ as (7).
  Compute $\{W^j\}_{j=0}^{N_s}$ using the word expansion function $ME(M^{(i)})$.
  for $r = 1, \cdots, Ns$ do
      Compute $T^r = F_r(T^{r-1}, W^{r-1})$.
  end for
  Compute $U^{N_s} = MA(T^{N_s}, W^{N_s})$.
  Assign and chaining variable array $CV^{(i+1)} \leftarrow U^{N_s}$ and ouput $CV^{(i+1)}$.
```

2.2 Brief Introduction on SHA-V

SHA-V is a PMD structural hash function family. It processes 1024-bit message blocks and produces a hash value of 128+32k ($k \in [0, 6]$) bits. The word length of SHA-V is 32 bits. The compression function of SHA-V consists of two SHA-1-like streams denoted by $_LCF$ and $_RCF$. Since we do not use any specific properties of $_LCF$ and $_RCF$, we refer interested readers to [7] for more details. The message hashing process of SHA-V consists of the following three stages:

Initialization: The given bit string message m is padded and cut into $t = \lceil \frac{|m|+1}{1024} \rceil$ 32-word message blocks, denoted as $M^{(0)}, \cdots, M^{(t-1)}$. Each message block $M^{(i)}$ is divided into two 512-bit sub-blocks $_LM^{(i)}$ and $_RM^{(i)}$ ($M^{(i)} = {_LM^{(i)}} \| {_RM^{(i)}}$ for $i \in [0, t-1]$). Two 5-word chaining variable arrays $_LCV^{(0)}$ and $_RCV^{(0)}$ are initialized to the constant initialization vectors $_LIV$ and $_RIV$ of SHA-V.

Compression: The two SHA-1-like compression functions works independently to update chaining variable arrays $_LCV^{(i)}$ and $_RCV^{(i)}$ ($i = 1, \cdots, t$) as

$$_LCV^{(i)} = {_LCF}({_LCV^{(i-1)}}, {_LM^{(i-1)}}) \tag{8}$$

$$_RCV^{(i)} = {_RCF}({_RCV^{(i-1)}}, {_RM^{(i-1)}}) \tag{9}$$

and acquiring the final chaining variable arrays $_LCV^{(t)}$ and $_RCV^{(t)}$.

Finalization: For $k \in [0, 6]$, the finalization function of SHA-V-($128 + 32k$), denoted by FV_k, return $(4 + k)$-word array $H = FV_k({_LCV^{(t)}}, {_RCV^{(t)}})$. Let

$$_LCV^{(t)} = ({_LV_0}, \cdots, {_LV_4}), \quad {_RCV^{(t)}} = ({_RV_0}, \cdots, {_RV_4}),$$

Then, the output array

$$H = (H_0, \cdots, H_{3+k})$$

can be computed according to Table 3.

Table 3. The Computation Methods of the $(4+k)$-Word Output Array $H = FV_k({}_LCV^{(t)}, {}_RCV^{(t)})$

k	H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9
0	${}_LV_0 + Y^{\lll 2}$	${}_LV_1 + Y^{\lll 3}$	${}_RV_0 + Z^{\lll 2}$	${}_RV_1 + Z^{\lll 3}$						
1	${}_LV_0 + {}_RV_0$	${}_LV_0 + {}_RV_1$	${}_LV_0 + {}_RV_2$	${}_LV_0 + {}_RV_3$						
2	${}_LV_0 + {}_RV_4^{\lll 2}$	${}_LV_1 + {}_RV_4^{\lll 3}$	${}_LV_2 + {}_RV_4^{\lll 5}$	${}_RV_0 + {}_RV_4^{\lll 2}$	${}_RV_1 + {}_RV_4^{\lll 3}$	${}_RV_2 + {}_RV_4^{\lll 5}$				
3	${}_LV_0 + {}_RV_4^{\lll 2}$	${}_LV_1 + {}_RV_4^{\lll 3}$	${}_LV_2 + {}_RV_4^{\lll 5}$	${}_LV_3 + {}_RV_4^{\lll 7}$	${}_LV_4 + {}_RV_4^{\lll 11}$	${}_RV_0 + {}_RV_4^{\lll 13}$	${}_RV_1 + {}_RV_4^{\lll 17}$			
4	${}_LV_0 + {}_RV_4^{\lll 2}$	${}_LV_1 + {}_RV_4^{\lll 3}$	${}_LV_2 + {}_RV_4^{\lll 5}$	${}_LV_3 + {}_RV_4^{\lll 7}$	${}_RV_0 + {}_LV_4^{\lll 2}$	${}_RV_1 + {}_LV_4^{\lll 3}$	${}_RV_2 + {}_LV_4^{\lll 5}$	${}_RV_3 + {}_LV_4^{\lll 7}$		
5	${}_LV_0 + {}_RV_4^{\lll 2}$	${}_LV_1 + {}_RV_4^{\lll 3}$	${}_LV_2 + {}_RV_4^{\lll 5}$	${}_LV_3 + {}_RV_4^{\lll 7}$	${}_LV_4 + {}_RV_4^{\lll 11}$	${}_RV_0 + {}_RV_4^{\lll 13}$	${}_RV_1 + {}_RV_4^{\lll 17}$	${}_RV_2 + {}_RV_4^{\lll 19}$	${}_RV_3 + {}_RV_4^{\lll 23}$	
6	${}_LV_0$	${}_LV_1$	${}_LV_2$	${}_LV_3$	${}_LV_4$	${}_RV_0$	${}_RV_1$	${}_RV_2$	${}_RV_3$	${}_RV_4$

$X = {}_RV_0 \oplus {}_RV_1 \oplus {}_RV_2 \oplus {}_RV_3 \oplus {}_RV_4$, $Y = {}_LV_2 \oplus {}_LV_3 \oplus {}_LV_4$, $Z = {}_RV_2 \oplus {}_RV_3 \oplus {}_RV_4$

2.3 Boomerang Attacks on Hash Functions

Resembling LSH, we consider the hash function HF defined as

$$H = HF(CV, M) = FIN_n \circ CF(CV, M) = FIN_n(U), \tag{10}$$

where CV is the chaining variable, M is the message block, CF is the compression function and FIN_n is the finalization function to produce an n-bit hash value. The goal of the boomerang attack aims at constructing four CV-M pairs

$$({}_0CV, {}_0M), ({}_1CV, {}_1M), ({}_2CV, {}_2M), ({}_3CV, {}_3M),$$

whose hash values ${}_iH = HF({}_iCV, {}_iM)$ $(i \in [0, 3])$ satisfying specific criteria within generic complexity bounds. According to different criteria, there are three types of boomerang attacks [18], denoted as Type I, II, III respectively. The hash array H defined in (10) has $|H| = n$ bits so the three types of boomerang attack can be defined as follows:

- Type I: Finding the quartet $({}_iCV, {}_iM)$ where $i \in [0, 3]$ satisfying ${}_0CV \oplus {}_2CV = {}_1CV \oplus {}_3CV = \alpha$ and ${}_0H \oplus {}_1H = {}_2H \oplus {}_3H = \delta$ for fixed predefined differences α and δ within the generic complexity bound 2^n.
- Type II: Finding the quartet $({}_iCV, {}_iM)$ where $i \in [0, 3]$ satisfying ${}_0H \oplus {}_1H = {}_2H \oplus {}_3H$ within the generic complexity bound $2^{n/3}$ [23]. This property is also called zero-sum or second-order differential collision.
- Type III: Finding the quartet $({}_iCV, {}_iM)$ where $i \in [0, 3]$ satisfying ${}_0CV \oplus {}_2CV = {}_1CV \oplus {}_3CV$ and ${}_0H \oplus {}_1H = {}_2H \oplus {}_3H$ within the generic complexity bound $2^{n/2}$.

As classical methods for boomerang attack, we review the known-related-key boomerang method given in [13], which we will use in Section 4. The iterative compression function CF in (10) can be regarded as a block cipher where the chaining variable CV is regarded as plaintext, the message M is regarded as key and the output of CF, denoted by U, is regarded as ciphertext. CF is decomposed into two sub-functions as $CF = CF_1 \circ CF_0$. In this way, we can start from the middle steps since CV and M can be chosen randomly [13,17]. Then we have a backward (top) differential characteristic $(\beta, \beta_k) \rightarrow \alpha$ with probability p for CF_0^{-1}, and a forward (bottom) differential characteristic $(\gamma, \gamma_k) \rightarrow \delta$ with probability q for CF_1. Finally, we can launch the known-related-key Type I boomerang attack with these two differential characteristics as follows:

1. Choose randomly a intermediate state $({}_0X, {}_0M)$ and compute $({}_iX, {}_iM), i = 2, 3, 4$ by ${}_2X = {}_0X \oplus \beta$, ${}_1X = {}_0X \oplus \gamma$, ${}_3X = {}_2X \oplus \gamma$, and ${}_2M = {}_0M \oplus \beta_k$, ${}_1M = M \oplus \gamma_k$, ${}_3M = {}_2M \oplus \gamma_k$.
2. Compute backward from $({}_iX, {}_iM)$ and obtain ${}_iCV$ by ${}_iCV = CF_0^{-1}({}_iX, {}_iM)$ $(i \in [0, 3])$.
3. Compute forward from $({}_iX, {}_iM)$ and obtain C_i by ${}_iU = CF_1({}_iX, {}_iM)$ $(i \in [0, 3])$.
4. Identify the conforming quartet $({}_iCV, {}_iM)$ for $i \in [0, 3]$ by checking whether ${}_0CV \oplus {}_2CV = {}_1CV \oplus {}_3CV = \alpha$ and ${}_0U \oplus {}_1U = {}_2U \oplus {}_3U = \delta$.

Type II and Type III boomerang attack only differs in the criteria in step 4.

Obviously, the boomerang attack described above is targeting at CF rather than HF in (10). But for LSH whose FIN_n function is linear, attacking CF is equivalent to attacking HF. This will be illustrated in detail later in Section 4.

3 The Structural Weakness of the LSH Hash Function

It is noticeable that in the compression functions of traditional MD structural hashes (such as SHA-1, SHA-2 etc.), there is a feeding forward operation

$$CF(CV, M) = E(CV, M) + CV \tag{11}$$

where the input CV is added to the output of a keyed permutation E. But, according to Section 2.1, the CF of LSH does not have feeding forward operations like (11). As can be seen in Algorithm 1, $CV^{(i+1)}$ is computed directly from the final state U^{N_s} without adding the previous chaining variable $CV^{(i)}$. In the finalization phase, as can be seen from (2), the hash value is linearly deduced from the final chaining variable $CV^{(t)}$ without involving previous $CV^{(0)}, \cdots, CV^{(t-1)}$. In the remainder of this paper, we show that the absence of feeding forward operation enables us to launch various attacks on full-round LSH hash functions with negligible complexities and 100% success probability.

3.1 Free-Start Collision Attack on Full LSH

For the hash function HF defined in (10) with output length $|H| = n$ bits, the free-start collision attack on hash function HF aims at finding two CV-M pairs $({}_0CV, {}_0M)$ and $({}_1CV, {}_1M)$ satisfying

$${}_0H = HF({}_0CV, {}_0M) = {}_1H = HF({}_1CV, {}_1M) \tag{12}$$

within a generic complexity bound $2^{n/2}$. Comparing with standard collision attack, the free-start collision does not have restrictions in the difference $\Delta_{in} = {}_0CV \oplus {}_1CV$ but the feeding forward strategy in (11) will force the adversary to balance both the output differences of CF and the input difference Δ_{in}, which can in turn increase difficulties.

Without the feeding forward operation, the LSH hash function is vulnerable to free-start collision attacks. A adversary can construct $({}_0CV, {}_0M)$ and $({}_1CV, {}_1M)$ satisfying (12) easily by taking the following steps:

1. Select a random 16-word array $U = (U_0, \cdots, U_{15})$;
2. Assign the arrays ${}_0U^{N_s} \leftarrow U$ and ${}_1U^{N_s} \leftarrow U$
3. Assign ${}_0M$ and ${}_1M$ with random values only satisfying ${}_0M \neq {}_1M$.
4. Compute $\{{}_0W^r\}_{r=0}^{N_s} = ME({}_0M)$ and $\{{}_1W^r\}_{r=0}^{N_s} = ME({}_1M)$;
5. With ${}_iU^{N_s}$ and $\{{}_iW^r\}_{r=0}^{N_s}$, compute backward to ${}_iT^0$ $(i = 0, 1)$;
6. Assign ${}_iCV \leftarrow {}_iT^0$ for $i = 0, 1$ and output $({}_0CV, {}_0M)$, $({}_1CV, {}_1M)$.

The $({}_0CV, {}_0M)$ and $({}_1CV, {}_1M)$ acquired above make a free-start collision for both CF and HF of LSH hash functions. Obviously, this attack only requires 2 queries of the LSH function to produce a free-start collision pair. As an MD structural hash function, the security of LSH relies heavily on the collision resistance of its compression function CF. Therefore, the existence of such an attack is inappropriate.

3.2 Pseudo-Preimage Attack on Full LSH

Still we have the hash function HF defined in (10) with output length $|H| = n$ bits. In the pseudo-preimage attack, the adversary can acquire a static output $\overline{H}$ of HF and he is supposed to find a pair $(\overline{CV}, \overline{M})$ satisfying $\overline{H} = HF(\overline{CV}, \overline{M})$ within the generic bound 2^n. Without the feeding forward operation (11), we can find a pseudo-preimage $(\overline{CV}, \overline{M})$ for any $\overline{H}$ by taking the following steps:

1. For a given 8-word array $\overline{H}$, we denote that

   $$\overline{H} = (\overline{H}_0, \cdots, \overline{H}_7),$$

   Select 8 random words $U_0, \cdots, U_7$ and construct a 16-word array

   $$\overline{U}^{N_s} = (U_0, \cdots, U_7, U_0 \oplus \overline{H}_0, \cdots, U_7 \oplus \overline{H}_7).$$

2. Assign $\overline{M}$ with random values and compute $\{\overline{W}^r\}_{r=0}^{N_s} = ME(\overline{M})$;
3. With $\overline{U}^{N_s}$ and $\{\overline{W}^r\}_{r=0}^{N_s}$, compute backward to $\overline{T}^0$;
4. Assign $\overline{CV} \leftarrow \overline{T}^0$ and output the pair $(\overline{CV}, \overline{M})$.

The acquired $(\overline{CV}, \overline{M})$ is a pseudo-preimage for both CF and HF. This attack only requires 1 query of LSH hash function, which largely challenges the security of LSH.

4 Boomerang Attacks on Round-Reduced LSH Hash Functions

In order to evaluate the strength of the LSH round function, we launch Type I and III boomerang attacks [1] on round-reduced LSH hash functions. We mainly describes the attack on LSH-512 and that of LSH-256 can be deduced accordingly.

Note: Many previous Type I boomerang results, such as [21,22], can only work on the compression functions (CF of (10)) or the keyed permutations (E of (11)) rather than the whole hash functions (HF of (10)). But for LSH, it is equivalent to attack CF, E and HF due to the absence of feeding forward operations and the linearity of the FIN_n finalization function.

4.1 Construction of Differential Characteristics

The very first step for the boomerang attack is constructing two differential characteristics with high probability. Since LSH is an ARX hash function family (only use three simple operations namely **Modular Add** "+", **Rotation** "$\ggg$" and **XOR**"$\oplus$"), we can use the XOR difference and deduce the difference linearly by considering the only nonlinear operation "+" as similar linear operation "$\oplus$".

The XOR difference in this paper is represented in two forms as follow:

- **Hex form:** such as Δv = 0x8003 indicates that bits $v[0, 1, 15]$ of the word v are active (having non-zero XOR difference).
- **Numeric form:** such as $\Delta v = (15, 1, 0)$ is equivalent to Δv = 0x8003 in hex form. Besides, if Δv = 0x0 in hex form, we denoted by $\Delta v = \phi$ in numeric form.

The two forms are used for presentation and linear deduction of the differential characteristics. For example, in the MX_r function of LSH, we have

$$aX = X + Y.$$

If we have acquired ΔX and ΔY, we can linearly deduce ΔaX as

$$\Delta aX = \Delta X \oplus \Delta Y.$$

Once we have determined the differences of the r-th chaining variable ΔT^r ($r \in [0, N_s]$ and two consecutive word arrays $\Delta W^{r'}$, $\Delta W^{r'+1}$ ($r' \in [0, N_s]$), we can linearly extend the difference backward and forward.

We construct the two differential characteristics for the boomerang attack, where the top differential characteristic is from round 0 to 8 and bottom differential difference is from 8 to 14. The differential characteristics are constructed based on the following observation.

Observation 1 *If we have $\Delta T^r = \Delta W^r \neq \phi$ and $\Delta W^{r+1} = \phi$, we can pass round r, $r + 1$ for free.*

Proof. Since $\Delta T^r = \Delta W^r \neq \phi$, we have $\Delta V^r = \phi$ after the first MA operation. Since $\Delta W^{r+1} = \phi$, no differences will be injected until the MA operation after T^{r+2}. □

We denote the difference of the top by $\Delta^t T^r$ ($r \in [0, 8]$ and that of the bottom by $\Delta^b T^r$ ($r \in [8, 14]$). Similarly, the difference for the word arrays are denoted as $\Delta^t W^r$ ($r \in [0, 8]$) in the top characteristic and $\Delta^b W^r$ ($r \in [8, 14]$) in the bottom characteristic. The main procedures for our characteristic construction can be summarized as follows:

Import Difference: We first import simple difference to message block $\Delta^t W^3$, $\Delta^t W^4$ ($\Delta^b W^{10}$, $\Delta^b W^{11}$) and the intermediate state $\Delta^t T^3$ ($\Delta^b T^{10}$).

Linear Extension: We linearly extend the difference backward to round 0 (8) and forward to round 8 (14) to acquire the whole top (bottom) differential characteristic.

Construct the Top Differential Characteristic: Based on Observation 1, we import the same 1-bit differences to both $\Delta^t W^3$ and $\Delta^t T_0^3$, and set $\Delta^t W^4 = \phi$. The differences are set as:

$$\Delta^t W_0^3 = \Delta^t T_0^3 = (63)$$

[1] Type II boomerang attacks on full-round LSH hash functions are no harder than constructing two free-start collisions in Section 3.1.

and $\Delta^t W_i^3 = \Delta^t T_i^3 = \phi$ ($r \in [1, 15]$). According to Observation 1, we can pass round 3 and 4 with probability 1. Furthermore, the selection of word array differences will keep the effect of message extension constantly linear so that we do not need to consider the effect of carries in all message expansions (ME). After determining $\Delta^t W^3$, $\Delta^t W^4$ and $\Delta^t T^3$, we linearly extend the difference backward to $\Delta^t T^0$ and forward to $\Delta^t T^8$. We present the top differential characteristic for LSH-512 in Table 4 of Appendix A.

The same strategy can also be carried out on LSH-256 where the differences are set similarly as

$$\Delta^t W_0^3 = \Delta^t T_0^3 = (31)$$

and $\Delta^t W_i^3 = \Delta^t T_i^3 = \phi$ ($i \in [1, 15]$). After linear extensions, we present the characteristic for LSH-256 in Table 5 of Appendix A.

Construct the Bottom Differential Characteristic: The strategy of constructing the bottom differential characteristic is similar to that of its top counterpart. We import 1-bit differences at W^{10} and T^{10}, and no difference at W^{11}. Following Observation 1, we assign that

$$\Delta^b W_y^{10} = \Delta^b T_y^{11} = (63) \tag{13}$$

and $\Delta^b W_i^{10} = \Delta^b T_i^{10} = \phi$ ($i \in [0, 15] \backslash \{y\}$). The selection of position y in (13) should meet the following criteria:

1. When linearly extend the difference from $\Delta^b T^{10}(y)$ to $\Delta^b T^8(y)$, make sure that
$$\Delta^t T_i^8 \wedge \Delta^b T_i^8(y) = 0\text{x}0, i \in [0, 15]. \tag{14}$$
2. When linearly extend the difference from $\Delta^b W^{10}(y)$ to $\Delta^b W^7(y)$, make sure that
$$\Delta^t W_i^r \wedge \Delta^b W_i^r(y) = 0\text{x}0, i \in [0, 15] \tag{15}$$
where $r = 7, 8, 9$.
3. When linearly extend the difference from $\Delta^b T^{10}(y)$ to $\Delta^b T^{14}(y)$, there is no contradicting bit conditions.

The restrictions (14) and (15) avoid the contradictions in the intersection part of the two differential characteristics. The 3rd restriction is to filter some inconsistent characteristics. Similar to SHA-2, the LSH round functions can cause many two-bit conditions so we use the method introduced by Mendel et al. in [24] to detect contradictions. The available ys compose a set $\mathbb{Y}_{512}$ defined as

$$\mathbb{Y}_{512} = \{5, 8, 9, 10, 11, 12, 13\}.$$

With the absence of feeding forward operation and linear FIN_n function of LSH, we can decide the differences of the final chaining variable U^{14} and the hash array H of the whole LSH hash functions as

$$\Delta^b U = (\Delta^b U_0^{14}, \cdots, \Delta^b U_{15}^{14}) = (\Delta^b T_0^{14} \oplus \Delta^b W_0^{14}, \cdots, \Delta^b T_{15}^{14} \oplus \Delta^b W_{15}^{14}), \tag{16}$$
$$\Delta^b H = (\Delta^b U_0^{14} \oplus \Delta^b U_8^{14}, \cdots, \Delta^b U_7^{14} \oplus \Delta^b U_{15}^{14}). \tag{17}$$

Since the operations $U^{14} = MA(T^{14}, M^{14})$ and $H = FIN_n(U^{14})$ are linear, we only need to consider the the procedure from T^0 to T^{14}.

For LSH-256, we assign that

$$\Delta^b W_y^{10} = \Delta^b T_y^{11} = (31)$$

and $\Delta^b W_i^{10} = \Delta^b T_i^{10} = \phi$ ($i \in [0, 15] \backslash \{y\}$). The available y candidates are limited to the elements in a set $\mathbb{Y}_{256} = \{8, 14\}$.

We set $y = 8 \in \mathbb{Y}_{512} \bigcap \mathbb{Y}_{256}$ and deduce an available bottom differential characteristic for LSH-512 in Table 6 and that for LSH-256 in Table 7 of Appendix A.

4.2 Finding the Boomerang Quartet Using Message Modification Technique

We give detailed description to the process of finding Type I boomerang quartets. Due to the similarities between Type I and Type III boomerang, we only illustrates their differences at the end of this section.

According to previous analysis, we only have to consider the quartets $({}_iCV, {}_iM)$ $(i \in [0,3])$ conforming the top and bottom characteristics from T^0 to T^{14}. Therefore, the goal of our boomerang attack is to find a quartet, denoted by $({}_0T^0, {}_1T^0, {}_2T^0, {}_3T^0)$, and the corresponding message blocks $({}_0W^r, {}_1W^r, {}_2W^r, {}_3W^r)$ $(r = 0, 1)$ that satisfy

$$ {}_0T^0 \oplus {}_2T^0 = {}_1T^0 \oplus {}_3T^0 = \Delta^t T^0 \tag{18}$$

and, after 14 rounds, the corresponding quartet $({}_0T^{14}, {}_1T^{14}, {}_2T^{14}, {}_3T^{14})$ satisfies

$$ {}_0T^{14} \oplus {}_1T^{14} = {}_2T^{14} \oplus {}_3T^{14} = \Delta^b T^{14}. \tag{19}$$

WA and WP are linear operations and do not generate any bit conditions. We only have to consider the effect of MX_j operations that connects the intermediate states U^j and V^j $(j \in [0, 13])$. Since our top and bottom characteristics intersect at round 8, we can construct available ${}_0U^8$, ${}_0V^7$, ${}_0V^6$ so that ${}_0T^8$, ${}_0W^7$, ${}_0W^8$ are determined accordingly. Once ${}_0T^8$ is settled, we can deduce ${}_1T^8$, ${}_2T^8$ and ${}_3T^8$ since

$$ {}_0T^8 \oplus {}_2T^8 = {}_1T^8 \oplus {}_3T^8 = \Delta^t T^8 \tag{20}$$

$$ {}_0T^8 \oplus {}_1T^8 = {}_2T^8 \oplus {}_3T^8 = \Delta^b T^8 \tag{21}$$

${}_1W^r$, ${}_2W^r$ and ${}_3W^r$ $(r = 7, 8)$ are decided since

$$ {}_0W^r \oplus {}_2W^r = {}_1W^r \oplus {}_3W^r = \Delta^t W^r, \tag{22}$$

$$ {}_0W^r \oplus {}_1W^r = {}_2W^r \oplus {}_3W^r = \Delta^b W^r. \tag{23}$$

The method of finding a Type I quartet is as follows:

Phase 1: Find an available starting point:

1. Construct an intermediate state, denoted by U^8 by setting the values of their 16 words randomly.
2. Compute forward to V^8 through MX_8. During the process, if one of bit conditions, which are deduced from the top characteristics, is violated, we can fix it by modifying the words U^8_j where $j \in \lambda_8 = \{1, 3, 5, 9, 11, 13\}$.
3. Construct an intermediate state, denoted by V^7 by setting the values of their 16 words randomly.
4. Compute backward to U^7 through MX_7^{-1} and compensate the corresponding bit conditions of the bottom characteristic by modifying V^7_j where $j \in \lambda_7 = \{0, 1, 2, 4, 6, 8, 9, 10, 12, 14\}$.
5. Construct an intermediate state, denoted by V^6 by setting the values of their 16 words randomly.
6. Compute backward to U^6 through MX_6^{-1} and compensate the corresponding bit conditions of the bottom characteristic by modifying V^6_j where $j \in \lambda_6 = \{0, 2, 6, 8, 10, 14\}$.

Phase 2: Find boomerang quartet:

7. With the knowledge of V^6 and U^7, we deduce $W^7 = V^7 \oplus WP(U^6)$.
8. With the knowledge of V^7 and U^8, we deduce $W^8 = V^8 \oplus WP(U^7)$.
9. After all conditions between round 7 and 8 are satisfied, we assign that ${}_0T^8 \leftarrow V^8 \oplus W^8$, ${}_0W^r \leftarrow W^r$ $(r = 7, 8)$. We also assign corresponding values to ${}_1T^8$, ${}_2T^8$, ${}_3T^8$ according to (20) (21) and to ${}_1W^r$, ${}_2W^r$, ${}_3W^r$ according to (22) (23).
10. Having acquired $({}_0T^8, {}_1T^8, {}_2T^8, {}_3T^8)$ and $({}_0W^r, {}_1W^r, {}_2W^r, {}_3W^r)$ for $r = 7, 8$, we compute backward to $({}_0T^0, {}_1T^0, {}_2T^0, {}_3T^0)$ and forward to $({}_0T^{14}, {}_1T^{14}, {}_2T^{14}, {}_3T^{14})$.
11. If $({}_0T^0, {}_1T^0, {}_2T^0, {}_3T^0)$ satisfies (18) and $({}_0T^{14}, {}_1T^{14}, {}_2T^{14}, {}_3T^{14})$ satisfies (19), output the quartet $({}_0T^8_0, {}_1T^8_0, {}_2T^8_0, {}_3T^8_0)$ and the corresponding $({}_0W^r, {}_1W^r, {}_2W^r, {}_3W^r)$ for $r = 0, 1$. Otherwise, do the following substeps:
    (a) Substitute the words V^7_j $(j \in [0, 15]\backslash\lambda_7)$ with new random values and recompute U^7.
    (b) Substitute V^6_j $(j \in [0, 15]\backslash\lambda_6)$ and U^8_i $(i \in [0, 15]\backslash\lambda_8)$ with new random values.
    (c) Go to Step 7.

After acquiring the above $({}_iT^0, {}_iW^0, {}_iW^1)$, we can assign ${}_iCV \leftarrow {}_iT^0$, ${}_iM \leftarrow {}_iW^0 \| {}_iW^1$ and the pairs $({}_iCV, {}_iM)$ $(i \in [0, 3])$ are boomerang quartets for the whole 14-round LSH hash functions. The hash arrays ${}_iH$ of $HF({}_iCV, {}_iM)$ $(i \in [0, 3])$ satisfy the difference in (17).

Complexity analysis for Type I Boomerang. For LSH-512, from $\Delta^t T^6$ to $\Delta^t T^8$, there are totally 117 bit-conditions and 101 of them can be fixed using message modification technique. In the bottom characteristic, 6 out of 7 conditions can be fixed in $\Delta^b T^8 \rightarrow \Delta^b T^9$. Therefore, the Phase 1 will take about

$2^{16+1} = 2^{17}$ queries to find an available starting point. Since there are 115 conditions in the remaining top characteristic and 39 in the bottom one, the complexity of Phase 2 is $2^{(115+39)\times 2} = 2^{308}$. So the overall complexity of the boomerang attack on LSH-512 is $2^{17} + 2^{308} = 2^{308}$.

For LSH-256, 99 out of 117 conditions in the middle can be fixed with message modification technique so the complexity for Phase 1 is 2^{18}. There are 82 conditions in the top characteristic and 39 in the bottom one, so the complexity of Phase 2 is $2^{(82+39)\times 2} = 2^{242}$ which is also the overall complexity.

Type III Boomerang. The only difference between Type I and Type III boomerang attacks on LSH occurs at step 11 where (18) and (19) are replaced respectively by

$$
{}_{0}T^{0} \oplus {}_{2}T^{0} = {}_{1}T^{0} \oplus {}_{3}T^{0}
$$
$$
{}_{0}T^{14} \oplus {}_{1}T^{14} = {}_{2}T^{14} \oplus {}_{3}T^{14}.
$$

According to [18], we evaluate the complexity of a Type III boomerang attack on 14-round LSH-512 hash function as $2^{17} + 3^{115+39} = 2^{244.1}$, only slightly lower than the generic bound 2^{256}. As to LSH-256, the complexity of the 14-round attack is $2^{18} + 3^{82+39} = 2^{191.8}$, exceeding the generic bound 2^{128}. Therefore, we can only start from T^1 of LSH-256 and acquire a 13-round Type III boomerang attack on LSH-256 with a complexity of $2^{98.3}$.

Note: With a linear LSH FIN_n function, the Type III boomerang would still be effective even if the feeding forward operations were adopted by LSH.

Practical Verifications. For LSH-512 and LSH-256, we find 11-round (from T^2 to T^{13}) boomerang Type I quartets satisfying our characteristics and present them in Table 8 and Table 9.

5 The Weaknesses of the SHA-V Hash Functions

As can be seen from Section 2.2, the two SHA-1-like streams ${}_{L}CF$ and ${}_{R}CF$ are processing independent 512-bit message blocks. We can use the divide-and-conquer strategy to find preimages and construct collisions.

5.1 Preimage Attacks on SHA-V

For a given $(128 + 32k)$-bit H $(k \in [0, 6])$, we can find a 2048-bit message M satisfying $H = SHAV(M)$ by taking the following steps:

Phase 1: Construct Lookup Tables:

1. For $p \in [0, 2^{80}]$, select different 512-bit message blocks ${}_{L}^{p}M^{(0)}$, compute the corresponding ${}_{L}^{p}CV^{(1)} = {}_{L}CF({}_{L}CV^{(0)}, {}_{L}^{p}M^{(0)})$ and store the pairs $({}_{L}^{p}M^{(0)}, {}_{L}^{p}CV^{(1)})$ in a table ${}_{L}\mathcal{T}$ sorted by ${}_{L}^{p}CV^{(1)}$.
2. For $q \in [0, 2^{80}]$, select different 512-bit message blocks ${}_{L}^{q}M^{(0)}$, compute the corresponding ${}_{R}^{q}CV^{(1)} = {}_{R}CF({}_{R}CV^{(0)}, {}_{R}^{q}M^{(0)})$ and store the pairs $({}_{R}^{q}M^{(0)}, {}_{R}^{q}CV^{(1)})$ in a table ${}_{R}\mathcal{T}$ sorted by ${}_{R}^{q}CV^{(1)}$.

Phase 2: Match in the Middle:

3. For the given H, construct two 5-word arrays
$$
{}_{L}V = ({}_{L}V_0, \cdots, {}_{L}V_4) \text{ and } {}_{R}V = ({}_{R}V_0, \cdots, {}_{R}V_4)
$$
satisfying $H = FV_k({}_{L}V, {}_{R}V)$ (for $k < 6$, there are multiple solutions and we pick one of them).
4. Construct a random 512-bit message block ${}_{L}M^{(1)}$ and compute the 5-word array ${}_{L}C = {}_{L}CF^{-1}({}_{L}V, {}_{L}M^{(1)})$.
5. Lookup in the table ${}_{L}\mathcal{T}$ and check whether there is a pair $({}_{L}^{p}M^{(0)}, {}_{L}^{p}CV^{(1)}) \in {}_{L}\mathcal{T}$ satisfying ${}_{L}^{p}CV^{(1)} = {}_{L}C$. If no matching is found, go back to step 4.
6. Construct a random 512-bit message block ${}_{R}M^{(1)}$ and compute the 5-word array ${}_{R}C = {}_{R}CF^{-1}({}_{R}V, {}_{R}M^{(1)})$.
7. Lookup in the table ${}_{R}\mathcal{T}$ and check whether there is a pair $({}_{R}^{q}M^{(0)}, {}_{R}^{q}CV^{(1)}) \in {}_{R}\mathcal{T}$ satisfying ${}_{R}^{q}CV^{(1)} = {}_{R}C$. If no matching is found, go back to step 6.

Phase 3: Construct the Target M:

8. Assign a 2048-bit message block $M \leftarrow {}_{L}^{p}M^{(0)} \| {}_{R}^{q}M^{(0)} \| {}_{L}M^{(1)} \| {}_{R}M^{(1)}$ and output M.

Complexity Analysis: In Phase 1, it takes 2^{80} queries of ${}_{L}CF$ to construct the lookup table ${}_{L}\mathcal{T}$ and another 2^{80} queries of ${}_{R}CF$ to construct the lookup table ${}_{R}\mathcal{T}$. Both ${}_{L}\mathcal{T}$ and ${}_{L}\mathcal{T}$ contains 2^{80} pairs so the time and memory complexity of Phase 1 are both $O(2^{80})$. In Phase 2, each matching requires about 2^{80} queries of ${}_{L}CF^{-1}$ (${}_{R}CF^{-1}$). So the time complexity of Phase 2 is also $O(2^{80})$. The memory complexity of this phase is negligible. The complexity of Phase 3 is $O(1)$. So the overall time complexity is dominated by Phase 2's $O(2^{80})$. The memory complexity is dominated by the size of ${}_{L}\mathcal{T}$ and ${}_{R}\mathcal{T}$, which is also $O(2^{80})$.

5.2 Collision Attacks on SHA-V

We first find a collision for ${}_LCF$. Since messages processed by ${}_RCF$ are independent ${}_LCF$, we can use identical message blocks in ${}_RCF$ stream. For a given $(128+32k)$-bit H, without utilizing any specific property of ${}_LCF$, we can find two 1024-bit messages M and M' satisfying $SHAV(M) = SHAV(M')$ by taking the following steps:

Phase 1: Find a Collision for ${}_LCF$.

1. For $p \in [0, 2^{80}]$, select different 512-bit message blocks ${}^p_LM$, compute the corresponding ${}^p_LCV^{(1)} = {}_LCF({}_LCV^{(0)}, {}^p_LM)$ and store the pairs $({}^p_LM, {}^p_LCV^{(1)})$ in a table ${}_L\mathcal{T}$ ordered by ${}^p_LCV^{(1)}$.
2. Construct a random 512-bit message block ${}_LM'$ and compute the 5-word array ${}_LC = {}_LCF({}_LCV^{(0)}, {}_LM')$.
3. Lookup in the table ${}_L\mathcal{T}$ and check whether there is a pair $({}^p_LM, {}^p_LCV^{(1)}) \in {}_L\mathcal{T}$ satisfying ${}^p_LCV^{(1)} = {}_LC$. If no matching is found, go back to step 2.

Phase 2: Construct the Target M and M':

4. Construct a random 512-bit message block ${}_RM$.
5. Assign two 1024-bit message blocks $M \leftarrow {}^p_LM \| {}_RM$ and $M' \leftarrow {}_LM' \| {}_RM$. Output M and M'.

Complexity Analysis: Similarly, the time complexity is Phase 2's $O(2^{80})$. The memory complexity is also $O(2^{80})$ which is the size of ${}_L\mathcal{T}$. Obviously, the M and M' acquired above is a collision for all versions of SHA-V. But the complexities $O(2^{80})$ have exceeded the generic bounds of SHA-V-128 and SHA-V-160. Therefore, this collision attack can only work on SHA-V-$(128+32k)$ for $k \in [2, 6]$.

Improved Attacks: According to [7], ${}_LCF$ branch is identical to SHA-1. Therefore, all existing collision attacks on SHA-1, such as [2,25,26,27,28], can be used to replace the redundant collision finding process of Phase 1. For example, if we replace Phase 1 with the collision attack of [27], we can find a 1024-bit ${}_LM = {}^0_LM \| {}^1_LM$ and ${}_LM' = {}^0_LM' \| {}^1_LM'$ satisfying ${}_LCF({}_LCV^{(0)}, {}_LM) = {}_LCF({}_LCV^{(0)}, {}_LM')$ with a time complexity $O(2^{61})$ and a negligible memory complexity. Then, we generate two random 512-bit blocks ${}^0_RM$ and ${}^1_RM$. The targeted M and M' are therefore assigned as

$$M \leftarrow {}^0_LM \| {}^0_RM \| {}^1_LM \| {}^1_RM$$
$$M' \leftarrow {}^0_LM' \| {}^0_RM \| {}^1_LM' \| {}^1_RM.$$

In this way, we can lower the time complexity to $O(2^{61})$ and the memory complexity becomes negligible. This improved attack can then be applied to all versions of SHA-V.

6 Conclusion

The round function of LSH is extremely strong so that our boomerang attacks can only mount to about 50% of the total LSH rounds. But the absence of the feeding forward operations in the LSH compression functions can cause serious insecurity. It enables the adversary to construct free start collisions and pseudo-preimages on full LSH with negligible complexities. Based on our analysis, we claim: no matter how well designed the round functions are, the omission of the feeding forward operation is irrational for the wide-pipe MD structural LSH. Besides, according to our boomerang results, the linear finalization phase of the LSH hash functions is also a latent danger. Therefore, we suggest that the designers should consider better finalizations for both the compression functions and the hash functions of LSH.

As to SHA-V, the XOR combiner of two independent hash functions has proved to be weak in [8]. Processing independent message blocks makes SHA-V even weaker. Since SHA-1 has been proved insecure, the secure basis of SHA-V does not exist anymore. Therefore, the applications of SHA-V should be avoided in any circumstances.

References

1. Wang, X., Yu, H.: How to break md5 and other hash functions. In: Advances in Cryptology–EUROCRYPT 2005. Springer (2005) 19–35
2. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full sha-1. In: Advances in Cryptology–CRYPTO 2005, Springer (2005) 17–36
3. Kayser, R.F.: Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register **72**(212) (2007) 62

4. Bertoni, G., Daemen, J., Peeters, M., Assche, G.: The keccak reference. Submission to NIST (Round 3) **13** (2011)
5. US Department of Commerce, N.: Guideline for implementing cryptography in the federal government. NIST SP - 800-21 2nd ed. (1999)
6. Kim, D.C., Hong, D., Lee, J.K., Kim, W.H., Kwon, D.: Lsh: A new fast secure hash function family. In: Information Security and Cryptology-ICISC 2014. Springer (2014) 286–313
7. HER, Y.S., SAKURAI, K.: Design and analysis of cryptographic hash function for the next generation. In: International Workshop on Informations & Electrical Engineering. (2002)
8. Leurent, G., Wang, L.: The sum can be weaker than each part. In Oswald, E., Fischlin, M., eds.: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Volume 9056 of Lecture Notes in Computer Science., Springer (2015) 345–367
9. Joux, A.: Multicollisions in iterated hash functions. application to cascaded constructions. Advances in Cryptology C CRYPTO 2004 (2004) 306–316
10. Wagner, D.: The boomerang attack. In: Fast Software Encryption, Springer (1999) 156–170
11. Biryukov, A., Nikolić, I., Roy, A.: Boomerang attacks on blake-32. In: Fast Software Encryption, Springer (2011) 218–237
12. Lamberger, M., Mendel, F.: Higher-order differential attack on reduced sha-256. IACR Cryptology ePrint Archive **2011** (2011) 37
13. Biryukov, A., Lamberger, M., Mendel, F., Nikolić, I.: Second-order differential collisions for reduced sha-256. In: Advances in Cryptology–ASIACRYPT 2011. Springer (2011) 270–287
14. Mendel, F., Nad, T.: Boomerang distinguisher for the simd-512 compression function. In: Progress in Cryptology–INDOCRYPT 2011. Springer (2011) 255–269
15. Sasaki, Y., Wang, L., Takasaki, Y., Sakiyama, K., Ohta, K.: Boomerang distinguishers for full has-160 compression function. In: Advances in Information and Computer Security. Springer (2012) 156–169
16. Sasaki, Y., Wang, L.: Distinguishers beyond three rounds of the ripemd-128/-160 compression functions. In: Applied Cryptography and Network Security, Springer (2012) 275–292
17. Leurent, G., Roy, A.: Boomerang attacks on hash function using auxiliary differentials. In: Topics in Cryptology–CT-RSA 2012. Springer (2012) 215–230
18. Yu, H., Chen, J., Wang, X.: The boomerang attacks on the round-reduced skein-512. In: Selected Areas in Cryptography, Springer (2013) 287–303
19. Kircanski, A., Shen, Y., Wang, G., Youssef, A.M.: Boomerang and slide-rotational analysis of the sm3 hash function. In: Selected Areas in Cryptography, Springer (2013) 304–320
20. Bai, D., Yu, H., Wang, G., Wang, X.: Improved boomerang attacks on sm3. In: Information Security and Privacy, Springer (2013) 251–266
21. Dongxia, B., Hongbo, Y., Gaoli, W., Xiaoyun, W.: Improved boomerang attacks on round-reduced sm3 and keyed permutation of blake-256. IET Information Security **9**(3) (2014) 167 – 178
22. Hao, Y.: The boomerang attacks on blake and blake2. In: Inscrypt. (2014) 286–310
23. Wagner, D.: A generalized birthday problem. In: Advances in cryptologyCRYPTO 2002. Springer (2002) 288–304
24. Mendel, F., Nad, T., Schläffer, M.: Finding sha-2 characteristics: Searching through a minefield of contradictions. In: Advances in Cryptology–ASIACRYPT 2011. Springer (2011) 288–307
25. X Wang, A C Yao, F.Y.: Cryptanalysis on sha-1 hash function. In proceeding of The Cryptographic hash workshop. National Institute of Standards and Technology (2005) 430–448
26. Cochran, M.: Notes on the wang et al. 263 sha-1 differential path. IACR (2008)
27. Vincent Rijmen, E.O.: Update on sha-1. Topics in Cryptology C CT-RSA 2005 (2005) 58–71
28. Stevens, M.: New collision attacks on sha-1 based on optimal joint local-collision analysis. Advances in Cryptology C EUROCRYPT 2013 (2013) 245–261

A The Differential Characteristics for LSH-512 and LSH-256

We present the top and bottom characteristics in numerical form. The column named "Cond" indecates the number of bit conditions in the corresponding round. The symbol "fxd" indicates the number of conditions fixed by with message modifications. We only present the differences from T^0 to T^{14}. The differences of the chaining variable U^{14} and the hash arrays H can be linearly deduced according to (16) and (17).

Table 4: The Top Characteristic for LSH-512.

r	$\Delta^t T^r$ & $\Delta^t W^r$	Cond	r	$\Delta^t T^r$ & $\Delta^t W^r$	Cond
	$\Delta^t T_1^0 : 52, 47, 31, 28, 24, 8, 5$		3	$\Delta^t T_0^3 : 63$	0
Continued on Next Page					

Following Previous Page					
0	$\Delta^t T_2^0 : 63, 60, 57, 55, 52, 37, 34, 9, 6$	95		$\Delta^t W_0^3 : 63$	
	$\Delta^t T_4^0 : 63, 40$		4	ϕ	0
	$\Delta^t T_5^0 : 44, 39, 37, 32, 31, 28, 24, 16, 9, 8, 5, 1$			ϕ	
	$\Delta^t T_6^0 : 62, 58, 52, 49, 47, 45, 44, 40, 29, 26, 22, 1$		5	ϕ	0
	$\Delta^t T_7^0 : 63, 40$			$\Delta^t W_2^5 : 63$	
	$\Delta^t T_9^0 : 52, 47, 31, 28$		6	$\Delta^t T_8^6 : 9, 6$	20 (17fxd)
	$\Delta^t T_{10}^0 : 60, 57, 55, 52, 37, 34, 32, 29$			$\Delta^t T_{14}^6 : 41$	
	$\Delta^t T_{12}^0 : 63$			$\Delta^t W_2^6 : 63$	
	$\Delta^t T_{13}^0 : 44, 39, 37, 32, 31, 28, 24$		7	$\Delta^t T_0^7 : 59, 36, 0$	97 (84fxd)
	$\Delta^t T_{14}^0 : 52, 49, 47, 45, 44, 40, 29, 26, 24, 22, 21, 17$			$\Delta^t T_6^7 : 35, 12$	
	$\Delta^t T_{15}^0 : 63$			$\Delta^t T_8^7 : 22, 17$	
	$\Delta^t W_2^0 : 63$			$\Delta^t T_9^7 : 32, 29, 27, 24, 4, 1$	
1	$\Delta^t T_1^1 : 63$	19		$\Delta^t T_{12}^7 : 27, 24, 4, 1$	
	$\Delta^t T_2^1 : 31, 28, 24$			$\Delta^t T_{14}^7 : 49$	
	$\Delta^t T_3^1 : 63$			$\Delta^t W_1^7 : 63$	
	$\Delta^t T_6^1 : 61, 57, 23, 20, 16, 0$			$\Delta^t W_2^7 : 63$	
	$\Delta^t T_7^1 : 63, 56$		8	$\Delta^t T_0^8 : 59, 56, 52, 45, 42, 22, 19$	-
	$\Delta^t T_{10}^1 : 31, 28$			$\Delta^t T_1^8 : 37, 31, 30, 27, 14, 8, 7, 4$	
	$\Delta^t T_{14}^1 : 61, 23, 20, 0$			$\Delta^t T_4^8 : 45, 42, 38, 35, 22, 19, 15, 12$	
	$\Delta^t T_{15}^1 : 63$			$\Delta^t T_6^8 : 62, 35, 28, 21$	
	$\Delta^t W_1^1 : 63$			$\Delta^t T_8^8 : 9, 6$	
	$\Delta^t W_3^1 : 63$			$\Delta^t T_9^8 : 46, 43, 32, 29, 27, 25, 24, 20, 10, 7, 5, 2$	
2	$\Delta^t T_3^2 : 63$	1		$\Delta^t T_{10}^8 : 42, 37, 36, 35, 32, 31, 30, 27, 14, 9, 8, 7, 6, 4$	
	$\Delta^t T_6^2 : 63, 40$			$\Delta^t T_{12}^8 : 46, 32, 27, 25, 20, 10, 5$	
	$\Delta^t T_{14}^2 : 63$			$\Delta^t T_{14}^8 : 41$	
	$\Delta^t W_3^2 : 63$			$\Delta^t T_{15}^8 : 58, 55, 53, 51, 50, 48, 46, 43, 30, 27, 25, 23, 20$	
				$\Delta^t W_2^8 : 63$	

Table 5: The Top Characteristic for LSH-256.

r	$\Delta^t T^r$ & $\Delta^t W^r$	Cond	r	$\Delta^t T^r$ & $\Delta^t W^r$	Cond
0	$\Delta^t T_1^0 : 30, 26, 23, 22, 18, 15, 1$	59	3	$\Delta^t T_0^3 : 31$	0
	$\Delta^t T_2^0 : 31, 28, 24, 22, 21, 13, 9, 7, 6$			$\Delta^t W_0^3 : 31$	
	$\Delta^t T_4^0 : 31, 2$		4	ϕ	0
	$\Delta^t T_5^0 : 30, 14, 9, 1$			ϕ	
	$\Delta^t T_6^0 : 30, 29, 21, 17, 16, 15, 14, 12, 10, 9, 4, 0$		5	ϕ	3
	$\Delta^t T_7^0 : 31, 2$			$\Delta^t W_2^5 : 31$	
	$\Delta^t T_9^0 : 30, 23, 22, 15$		6	$\Delta^t T_8^6 : 21, 4$	20 (17fxd)
	$\Delta^t T_{10}^0 : 25, 24, 22, 21, 10, 9, 7, 6$			$\Delta^t T_{14}^6 : 5$	
	$\Delta^t T_{12}^0 : 31$			$\Delta^t W_2^6 : 31$	
	$\Delta^t T_{13}^0 : 30, 14, 9$		7	$\Delta^t T_0^7 : 6, 3, 2$	91 (76fxd)
	$\Delta^t T_{14}^0 : 30, 29, 18, 17, 15, 14, 13, 12, 10, 9, 1, 0$			$\Delta^t T_6^7 : 14, 11$	
	$\Delta^t T_{15}^0 : 31$			$\Delta^t T_8^7 : 29, 28$	
	$\Delta^t W_2^0 : 31$			$\Delta^t T_9^7 : 22, 19, 18, 5, 2, 1$	
1	$\Delta^t T_1^1 : 31$	19		$\Delta^t T_{12}^7 : 22, 19, 5, 2$	
	$\Delta^t T_2^1 : 30, 15, 10$			$\Delta^t T_{14}^7 : 13$	
	$\Delta^t T_3^1 : 31$			$\Delta^t W_1^7 : 31$	
	$\Delta^t T_6^1 : 26, 23, 21, 18, 9, 6$			$\Delta^t W_2^7 : 31$	
	$\Delta^t T_7^1 : 31, 26$		8	$\Delta^t T_0^8 : 30, 19, 18, 16, 4, 3, 1$	-
	$\Delta^t T_{10}^1 : 30, 15$			$\Delta^t T_1^8 : 22, 19, 12, 10, 9, 4$	
	$\Delta^t T_{14}^1 : 26, 23, 9, 6$			$\Delta^t T_4^8 : 31, 28, 19, 16, 14, 11, 4, 1$	
	$\Delta^t T_{15}^1 : 31$			$\Delta^t T_6^8 : 12, 11, 9, 6$	
	$\Delta^t W_1^1 : 31$			$\Delta^t T_8^8 : 21, 4$	
	$\Delta^t W_3^1 : 31$			$\Delta^t T_9^8 : 28, 25, 24, 19, 18, 14, 13, 11, 8, 7, 2, 1$	
2	$\Delta^t T_3^2 : 31$	1		$\Delta^t T_{10}^8 : 22, 21, 19, 18, 12, 10, 9, 8, 6, 3$	
	$\Delta^t T_6^2 : 31, 2$			$\Delta^t T_{12}^8 : 28, 25, 24, 19, 18, 14, 13$	
	$\Delta^t T_{14}^2 : 31$			$\Delta^t T_{14}^8 : 5$	
	$\Delta^t W_3^2 : 31$			$\Delta^t T_{15}^8 : 31, 30, 29, 27, 26, 20, 17, 16, 15, 12, 11, 3, 0$	
				$\Delta^t W_2^8 : 31$	

Table 6: The Bottom Characteristic for LSH-512.

r	$\Delta^b T^r$ & $\Delta^b W^r$	Cond	r	$\Delta^b T^r$ & $\Delta^b W^r$	Cond
8	$\Delta^b T_1^8 : 63, 40$	7 (6fxd)	12	ϕ	4
	$\Delta^b T_3^8 : 63, 40$			$\Delta^b W_{10}^{12} : 63$	
	$\Delta^b T_5^8 : 63, 56, 40, 33$		13	$\Delta^b T_8^{13} : 58, 22, 17$	34
	$\Delta^b T_{13}^8 : 63, 56$			$\Delta^b T_{14}^{13} : 49, 26$	
	$\Delta^b W_9^8 : 63$			$\Delta^b W_{10}^{13} : 63$	
	$\Delta^b W_{11}^8 : 63$		14	$\Delta^b T_0^{14} : 59, 56, 52, 36, 33, 29$	-
9	$\Delta^b T_2^9 : 63, 56$	1		$\Delta^b T_6^{14} : 35, 28, 12, 5$	
	$\Delta^b T_{10}^9 : 63$			$\Delta^b T_8^{14} : 9, 6, 2$	
	$\Delta^b T_{11}^9 : 63$			$\Delta^b T_9^{14} : 61, 32, 29, 27, 25, 24, 20, 4, 1$	
Continued on Next Page					

Following Previous Page					
	$\Delta^b W^9_{11}$: 63			$\Delta^b T^{14}_{12}$: 61, 32, 27, 25, 20, 4	
10	$\Delta^b T^{10}_8$: 63	0		$\Delta^b T^{14}_{14}$: 41, 34	
	$\Delta^b W^{10}_8$: 63			$\Delta^b W^{14}_9$: 63	
11	ϕ	0		$\Delta^b W^{14}_{10}$: 63	
	ϕ				

Table 7: The Bottom Characteristic for LSH-256.

r	$\Delta^b T^r$ & $\Delta^b W^r$	Cond	r	$\Delta^b T^r$ & $\Delta^b W^r$	Cond
8	$\Delta^b T^8_1$: 31, 2	7 (6fxd)	12	ϕ	4
	$\Delta^b T^8_3$: 31, 2			$\Delta^b W^{12}_{10}$: 31	
	$\Delta^b T^8_5$: 31, 29, 26, 2		13	$\Delta^b T^{13}_8$: 29, 28, 0	34
	$\Delta^b T^8_{13}$: 31, 26			$\Delta^b T^{13}_{14}$: 16, 13	
	$\Delta^b W^8_9$: 31			$\Delta^b W^{13}_{10}$: 31	
	$\Delta^b W^8_{11}$: 31		14	$\Delta^b T^{14}_0$: 30, 21, 18, 6, 3, 1	-
9	$\Delta^b T^9_2$: 31, 26	1		$\Delta^b T^{14}_6$: 14, 11, 9, 6	
	$\Delta^b T^9_{10}$: 31			$\Delta^b T^{14}_8$: 21, 16, 4	
	$\Delta^b T^9_{11}$: 31			$\Delta^b T^{14}_9$: 22, 19, 18, 17, 14, 13, 5, 2, 1	
	$\Delta^b W^9_{11}$: 31			$\Delta^b T^{14}_{12}$: 22, 19, 18, 17, 14, 13	
10	$\Delta^b T^{10}_8$: 31	0		$\Delta^b T^{14}_{14}$: 5, 0	
	$\Delta^b W^{10}_8$: 31			$\Delta^b W^{14}_9$: 31	
11	ϕ	0		$\Delta^b W^{14}_{10}$: 31	
	ϕ				

B Practical 11-Round Quartets for LSH-512 and LSH-256

The quartets presented are from T^2 to T^{13}.

Table 8: Boomerang Quartet for 11=Round LSH-512

$\Delta^t T^2_0$	$\Delta^t T^2_3$: 63, $\Delta^t T^2_6$: 63, 40, $\Delta^t T^2_{14}$: 63			
${}_0T^2$	0xd99b277f3ebb6338	0x4e831f7f7dd6e2e	0x68224d2e0db459ba	0xa3020a6b44a6f1f7
	0x17181dc3be2384da	0x27da6e1e043a7031	0x14513685c29b312a	0xc295923f1de60c5b
	0xf8141d272b3e8f5	0x930f419f7ff7bbd	0xb02dd0fbf989245b	0xc5d0d89608e93d58
	0xca33cbad8791f290	0x5fede51cae7a8ee1	0xaccd361f47708329	0x7a91aefeaa56c946
${}_1T^2$	0x7a35ee6527fdb787	0x9d4dd237dc6cceb	0xda4adae0aff0253f	0x2c45e768aad6f43c
	0x742f2725d5a5bba	0xe6eb4cc792da2a7	0x78d6c675d1dd6523	0x67c6afec8d1eacdd
	0xa90665ccb1e96458	0x9c7d530191f1beb2	0x1aea0bbc520dcb12	0x6556dc2f9a6de783
	0xa472425aa3a35e50	0x979b79774b50c074	0xc81034a6befaa93d	0xf2fc1e528767be1d
${}_2T^2$	0xd99b277f3ebb6338	0x4e831f7f7dd6e2e	0x68224d2e0db459ba	0x23020a6b44a6f1f7
	0x17181dc3be2384da	0x27da6e1e043a7031	0x94513785c29b312a	0xc295923f1de60c5b
	0xf8141d272b3e8f5	0x930f419f7ff7bbd	0xb02dd0fbf989245b	0xc5d0d89608e93d58
	0xca33cbad8791f290	0x5fede51cae7a8ee1	0x2ccd361f47708329	0x7a91aefeaa56c946
${}_3T^2$	0x7a35ee6527fdb787	0x9d4dd237dc6cceb	0xda4adae0aff0253f	0xac45e768aad6f43c
	0x742f2725d5a5bba	0xe6eb4cc792da2a7	0xf8d6c775d1dd6523	0x67c6afec8d1eacdd
	0xa90665ccb1e96458	0x9c7d530191f1beb2	0x1aea0bbc520dcb12	0x6556dc2f9a6de783
	0xa472425aa3a35e50	0x979b79774b50c074	0x481034a6befaa93d	0xf2fc1e528767be1d
$\Delta^t W^2$	$\Delta^t W^2_3$: 63			
${}_0W^2$	0x4be167a06d060782	0x1d74b2e3d560a003	0x58f4e907b0cca556	0x20d1ad09dec4916a
	0xf215b8df5de4048e	0x2701139e697027a7	0x74aefa306a1d7f79	0x47655056238a3b8b
	0xe5746e16789c2f00	0xf1767533875eeb4c	0x9d196098439f65b6	0xee3b3cbf47ae385
	0xb55b39daf96ab84e	0xbd418c7a94aa8cfa	0x360f65449b0fdc5d	0xc32e94b958453ceb
${}_1W^2$	0x4be167a06d060782	0x1d74b2e3d560a003	0x58f4e907b0cca556	0x20d1ad09dec4916a
	0xf215b8df5de4048e	0x2701139e697027a7	0x74aefa306a1d7f79	0x47655056238a3b8b
	0xe5746e16789c2f00	0x71767533875eeb4c	0x1d196098439f65b6	0xee3b3cbf47ae385
	0xb55b39daf96ab84e	0xbd418c7a94aa8cfa	0x360f65449b0fdc5d	0xc32e94b958453ceb
${}_2W^2$	0x4be167a06d060782	0x1d74b2e3d560a003	0x58f4e907b0cca556	0xa0d1ad09dec4916a
	0xf215b8df5de4048e	0x2701139e697027a7	0x74aefa306a1d7f79	0x47655056238a3b8b
	0xe5746e16789c2f00	0xf1767533875eeb4c	0x9d196098439f65b6	0xee3b3cbf47ae385
	0xb55b39daf96ab84e	0xbd418c7a94aa8cfa	0x360f65449b0fdc5d	0xc32e94b958453ceb
${}_3W^2$	0x4be167a06d060782	0x1d74b2e3d560a003	0x58f4e907b0cca556	0xa0d1ad09dec4916a
	0xf215b8df5de4048e	0x2701139e697027a7	0x74aefa306a1d7f79	0x47655056238a3b8b
	0xe5746e16789c2f00	0x71767533875eeb4c	0x1d196098439f65b6	0xee3b3cbf47ae385
	0xb55b39daf96ab84e	0xbd418c7a94aa8cfa	0x360f65449b0fdc5d	0xc32e94b958453ceb
$\Delta^t W^3$	$\Delta^t W^3_0$: 63			
${}_0W^3$	0x18893895e3264d	0xfe2e0404b56aabcd	0x8c48493a1820a213	0x61caf3da1af1b044
	0xca92045efaaac69e	0xa62b54423297527b	0x4f40ead3b36aece0	0xdb3894d352f891b6
	0xcc5750c3ab3a88fb	0x71275d3c93d832c0	0x6e3e000a676075ef	0xbb8cd78aa0c0c3a9
	0xa87984d2cfb45bd0	0x25ad76fee66d83e9	0x9c6fe3bf68bb2036	0x34ba020c52133250
${}_1W^3$	0x18893895e3264d	0xfe2e0404b56aabcd	0x8c48493a1820a213	0x61caf3da1af1b044
	0xca92045efaaac69e	0xa62b54423297527b	0x4f40ead3b36aece0	0xdb3894d352f891b6
	0xcc5750c3ab3a88fb	0x71275d3c93d832c0	0xee3e000a676075ef	0xbb8cd78aa0c0c3a9
	0xa87984d2cfb45bd0	0x25ad76fee66d83e9	0x9c6fe3bf68bb2036	0x34ba020c52133250
${}_2W^3$	0x8018893895e3264d	0xfe2e0404b56aabcd	0x8c48493a1820a213	0x61caf3da1af1b044
	0xca92045efaaac69e	0xa62b54423297527b	0x4f40ead3b36aece0	0xdb3894d352f891b6
	0xcc5750c3ab3a88fb	0x71275d3c93d832c0	0x6e3e000a676075ef	0xbb8cd78aa0c0c3a9
	0xa87984d2cfb45bd0	0x25ad76fee66d83e9	0x9c6fe3bf68bb2036	0x34ba020c52133250
${}_3W^3$	0x8018893895e3264d	0xfe2e0404b56aabcd	0x8c48493a1820a213	0x61caf3da1af1b044
	0xca92045efaaac69e	0xa62b54423297527b	0x4f40ead3b36aece0	0xdb3894d352f891b6
	0xcc5750c3ab3a88fb	0x71275d3c93d832c0	0xee3e000a676075ef	0xbb8cd78aa0c0c3a9
	0xa87984d2cfb45bd0	0x25ad76fee66d83e9	0x9c6fe3bf68bb2036	0x34ba020c52133250
Continued on Next Page				

Following Previous Page				
$\Delta^b T_0^{13}$	$\Delta^b T_8^{13}$: 58, 22, 17, $\Delta^b T_{14}^{13}$: 49, 26			
${}_0T^{13}$	0x8a8fc4f3e0d5021b	0xc41340867c8ab220	0xd64f491b8729ef89	0xdba3643b2ebb99ef
	0x514272156518d30a	0x360e71addf8ec86d	0xd1f1ce691e344ca4	0x39d23c603792f313
		0x8b794a1b03d0088d	0x2db9a168b66f6feb	0xfb5dbeaad406f8dd
	0xf91ca3887a27807	0x9b809c23cdb40f0c	0xac8097ef4bcad1dd	0xb6f82205a0f3d599
${}_1T^{13}$	0x8a8fc4f3e0d5021b	0xc41340867c8ab220	0xd64f491b8729ef89	0xdba3643b2ebb99ef
	0x514272156518d30a	0x360e71addf8ec86d	0xd1f1ce691e344ca4	0x39d23c603792f313
	0x759b7f1f98c3d518	0x8b794a1b03d0088d	0x2db9a168b66f6feb	0xfb5dbeaad406f8dd
	0xf91ca3887a27807	0x9b809c23cdb40f0c	0xac8297ef4fcad1dd	0xb6f82205a0f3d599
${}_2T^{13}$	0x486712399b031c93	0xe027fa1bd293822a	0x1253b1bd048fe1d0	0xa253edd40903ce19
	0xe9a141f8480e8994	0xb999eaba1b37c60	0xe70491ad915354be	0x90dea0b0534af7f5
	0xa57ddaacac67c7b7	0x1e8f8ce41c5eef82	0xa430a0c76dfd5874	0x98b1e35d40c4a4d6
	0x2bfc387b8a249b82	0xe682a22ee1762278	0x879bc860854c7978	0xcfe51005cc37bcc3
${}_3T^{13}$	0x486712399b031c93	0xe027fa1bd293822a	0x1253b1bd048fe1d0	0xa253edd40903ce19
	0xe9a141f8480e8994	0xb999eaba1b37c60	0xe70491ad915354be	0x90dea0b0534af7f5
	0xa17ddaacac25c7b7	0x1e8f8ce41c5eef82	0xa430a0c76dfd5874	0x98b1e35d40c4a4d6
	0x2bfc387b8a249b82	0xe682a22ee1762278	0x8799c860814c7978	0xcfe51005cc37bcc3

Table 9: Boomerang Quartet for 11=Round LSH-256

$\Delta^t T_0^2$	$\Delta^t T_3^2$: 31, $\Delta^t T_6^2$: 31, 2, $\Delta^t T_{14}^2$: 31							
${}_0T^2$	0xeab33b0c	0xc51e16f9	0x4be9f458	0x8081c224	0xcab7e3cf	0xc0ab11fc	0xb223bd49	0x51cacb7f
	0xf98405a1	0xaca1e3f9	0xff55a176	0xef293d6d	0x8fd140ee	0xbd932550	0x145a46f9	0xf0af205c
${}_1T^2$	0x87bca105	0x7612c4e6	0x59dc17e4	0x56fa0106	0xb709ef49	0xd450a842	0x29c16b78	0xe27b8747
	0xb062a004	0x8badd182	0x68e709a4	0xcf00c0c6	0x6ccabc61	0xf3f11347	0x550d4bc8	0xe179b345
${}_2T^2$	0xeab33b0c	0xc51e16f9	0x4be9f458	0x81c224	0xcab7e3cf	0xc0ab11fc	0x3223bd4d	0x51cacb7f
	0xf98405a1	0xaca1e3f9	0xff55a176	0xef293d6d	0x8fd140ee	0xbd932550	0x945a46f9	0xf0af205c
${}_3T^2$	0x87bca105	0x7612c4e6	0x59dc17e4	0xd6fa0106	0xb709ef49	0xd450a842	0xa9c16b7c	0xe27b8747
	0xb062a004	0x8badd182	0x68e709a4	0xcf00c0c6	0x6ccabc61	0xf3f11347	0xd50d4bc8	0xe179b345
$\Delta^t W^2$	$\Delta^t W_3^2$: 31							
${}_0W^2$	0xf58e43b7	0x8dd33165	0xe12e5fcd	0x74a7381d	0xa3555aad	0x5bf81153	0x4529b45	0x1400151f
	0x12662d67	0x53c8591	0x3e79ea10	0xc307ca7d	0x55df6002	0x67c75a52	0x1efde7c2	0x9bc17c2d
${}_1W^2$	0xf58e43b7	0x8dd33165	0xe12e5fcd	0x74a7381d	0xa3555aad	0x5bf81153	0x4529b45	0x1400151f
	0x12662d67	0x853c8591	0xbe79ea10	0xc307ca7d	0x55df6002	0x67c75a52	0x1efde7c2	0x9bc17c2d
${}_2W^2$	0xf58e43b7	0x8dd33165	0xe12e5fcd	0xf4a7381d	0xa3555aad	0x5bf81153	0x4529b45	0x1400151f
	0x12662d67	0x53c8591	0x3e79ea10	0xc307ca7d	0x55df6002	0x67c75a52	0x1efde7c2	0x9bc17c2d
${}_3W^2$	0xf58e43b7	0x8dd33165	0xe12e5fcd	0xf4a7381d	0xa3555aad	0x5bf81153	0x4529b45	0x1400151f
	0x12662d67	0x853c8591	0xbe79ea10	0xc307ca7d	0x55df6002	0x67c75a52	0x1efde7c2	0x9bc17c2d
$\Delta^t W^3$	$\Delta^t W_0^3$: 31							
${}_0W^3$	0xe2a658ff	0x7b058658	0xd2036a64	0x68d4c801	0x65cc0b24	0xff01e0eb	0xcbb99fa5	0x8ed9f2a6
	0xedf4a15	0x53641452	0x7e47e019	0x343fc3f6	0x43315557	0x1c316198	0xa8661a6b	0x63cd700a
${}_1W^3$	0xe2a658ff	0x7b058658	0xd2036a64	0x68d4c801	0x65cc0b24	0xff01e0eb	0xcbb99fa5	0x8ed9f2a6
	0xedf4a15	0x53641452	0xfe47e019	0x343fc3f6	0x43315557	0x1c316198	0xa8661a6b	0x63cd700a
${}_2W^3$	0x62a658ff	0x7b058658	0xd2036a64	0x68d4c801	0x65cc0b24	0xff01e0eb	0xcbb99fa5	0x8ed9f2a6
	0xedf4a15	0x53641452	0x7e47e019	0x343fc3f6	0x43315557	0x1c316198	0xa8661a6b	0x63cd700a
${}_3W^3$	0x62a658ff	0x7b058658	0xd2036a64	0x68d4c801	0x65cc0b24	0xff01e0eb	0xcbb99fa5	0x8ed9f2a6
	0xedf4a15	0x53641452	0xfe47e019	0x343fc3f6	0x43315557	0x1c316198	0xa8661a6b	0x63cd700a
$\Delta^b T_0^{13}$	$\Delta^b T_8^{13}$: 29, 28, 0, $\Delta^b T_{14}^{13}$: 16, 13							
${}_0T^{13}$	0xdecda23d	0x74edce2b	0x1bfeea03	0xdb18a03	0xcaf3ea3d	0xb7ba87ca	0xf8753736	0xa78c911e
	0x41db0b33	0xdf99ff5d	0x70685013	0xfd71be34	0x42e77e95	0x46ce6177	0xb7d5d759	0x9dad9ea8
${}_1T^{13}$	0xdecda23d	0x74edce2b	0x1bfeea03	0xdb18a03	0xcaf3ea3d	0xb7ba87ca	0xf8753736	0xa78c911e
	0x71db0b32	0xdf99ff5d	0x70685013	0xfd71be34	0x42e77e95	0x46ce6177	0xb7d4f759	0x9dad9ea8
${}_2T^{13}$	0x8e2c6754	0x23b7fca0	0x2cb78e4b	0xb469e132	0x7177152b	0x885463ec	0x1fae931	0xe0381d3c
	0x413063a4	0xe366cc8c	0xa6971d8f	0x14ae711	0xd020556d	0xafb55a0e	0x6d8ad9e2	0xa1f9eb7b
${}_3T^{13}$	0x8e2c6754	0x23b7fca0	0x2cb78e4b	0xb469e132	0x7177152b	0x885463ec	0x1fae931	0xe0381d3c
	0x713063a5	0xe366cc8c	0xa6971d8f	0x14ae711	0xd020556d	0xafb55a0e	0x6d8bf9e2	0xa1f9eb7b