Attacks on JH, Grøstl and SMASH Hash Functions

Yiyuan Luo and Xuejia Lai

luoyiyuan@gmail.com

Abstract. JH and Grøstl hash functions are two of the five finalists in NIST SHA-3 competition. JH-s and Grøstl-s are based on a $2n$ bit compression function and the final output is truncated to s bits, where n is 512 and s can be 224,256,384 and 512. Previous security proofs show that JH-s and Grøstl-s are optimal collision resistance without length padding to the last block.

In this paper we present collision and preimage attacks on JH-s and Grøstl-s without length padding to the last block. For collision attack on JH-s, after a $\frac{1}{e}2^{s/4+n}$ precomputing, the adversary needs $2^{s/4}$ queries to the underlying compression function to find a new collision. For preimage attack on JH-s, after a $\frac{1}{e}2^{s/2+n}$ precomputing, the adversary needs $2^{s/2}$ queries to the underlying compression function to find a new preimage. If $s = 224$, the attacker only needs 2^{57} and 2^{113} compression function queries to mount a new collision attack and preimage attack respectively. For Grøstl, the query complexity of our collision and preimage attack are one half of birthday collision attack and exhaustive preimage attack respectively.

We also discuss how our attack works when the length is padded to the last message block. Our attacks exploit structure flaws in the design of JH and Grøstl. It is easily applied to MJH and SMASH and other generalizations since they have similar structure (we call it Even-Mansour structure). At the same time the provable security of chopMD in the literature is challenged. Through our attack, it is easy to see that the chopMD mode used in JH or Grøstl does not improve its security.

1 Introduction

Cryptographic hash function is one of the most important primitives in cryptography [17]. A hash function maps from message of arbitrary length to a fixed length. A hash function usually consists of iteration of a compression function. One first designs a fixed domain compression function and then extends the domain to an arbitrary domain by iterating that function.

Since some popular hash functions such as MD5 [20] and SHA-1 [8] have been attacked [23,22], NIST has launched a competition for a new hash function standard SHA-3. JH and Grøstl are two of the five finalists in the SHA-3 competition [24,10]. As the sponge construction [2], JH's compression function uses a single large fixed $2n$-bit permutation, then the chopMD [5] domain extension is applied to the compression function. Grøstl is similar as JH except

that its compression function uses two $2n$-bit permutations and a final output transformation is applied to the chain value to get the hash value.

The hash value is the last s bits of the output of the last block compression function. In the design of JH and Grøstl, n is 512 and s can be 224,256,384 and 512. After JH, Lee and Stam proposed a variant MJH based on a an (n, n) blockcipher [16]. The hash value of MJH is $2n$ bits. Before Grøstl, SMASH is proposed by Knudsen in 2005 and uses one permutation [12]. After that, Pramstaller *et al.* gave collision attacks and Lamerger *et al.* gave second preimage attacks on it [19,13,14]. Knudsen suggested two tweaked versions of SMASH to thwart the attack. Later Fouque *et al.* proposed attacks on the two tweaked versions and proposed a generalization of SMASH using two permutations [9].

Previous security results for JH and MJH. In the provable security literature, the underlying primitives are assume to be ideal, thus the fixed permutation is assumed to be an ideal permutation. The first provable security result for the mode of JH is its indifferentiability[4]. Bhattacharyya *et al.* proved that JH-s is indifferentiable from a random oracle up to $O(2^{n/3})$ queries to the ideal permutation when $s \leq n$.

In [15], Lee and Hong proved that JH-s without length padding to the last block is collision resistance up to $O(2^{s/2})$ queries and claimed that JH-s is optimal collision resistance in the ideal permutation model when $s \leq n$. The proof does not require the length of the message is padded at the end of the message.

In [18], Moody *et al.* improve the indifferentiability security bound for the JH mode to $O(2^{n/3})$ queries to the ideal permutation when $s \leq n$.

In [16], the designers proved that MJH without length padding to the last block is collision resistance up to $O(2^{\frac{2n}{3}-\log n})$ queries.

In [11], Hong and Kwon make a collision attack with time complexity 2^{124} on MJH for $n = 128$ and the preimage attack with time complexity $2^{3n/2+2}$.

Previous security results for Grøstl and SMASH. The first provable security result for the mode of Grøstl is its indifferentiability[1]. Andreeva *et al.* proved that Grøstl-s is indifferentiable from a random oracle up to $O(2^{s/2})$ queries to the ideal permutations when $s \leq n$ and the chain value is $2n$ bits. The proof does not require the message length is padded at the end of the message.

After the proposal of SMASH by Knudsen [12], Pramstaller *et al.* gave collision attacks and Lamerger *et al.* gave second preimage attacks on it [19,13,14]. Knudsen suggested two tweaked versions of SMASH to thwart the attack. Later Fouque *et al.* proposed attacks on the two tweaked versions and proposed a generalization of SMASH using two permutations [9]. This generalization is proved secure against collision and preimage attack in the ideal permutation model in $\Omega(2^{n/2})$ queries and $\Omega(2^{n/4})$ respectively, where the hash value is $2n$ bits. Fouque *et al.* also proposed a non-trival collision attack in $\Omega(2^{3n/8})$ queries.

Our contribution. In the compression function of JH, Grøstl and SMASH, first a transformation is applied to the message block , later it is (xor) added to both the input and output of the permutation to get the chain value. We

call it Evan-Mansour structure [7,6]. While in Sponge compression function, the message block is only added to the input of the permutation. Our attacks actually are based on this observation.

In this paper we present collision and preimage attacks on JH-s and Grøstl-s. For collision attack on JH-s, after a $\frac{1}{e}2^{(s+l)/4+n}$ precomputing, the adversary needs $2^{(s+l)/4}$ queries to the underlying compression function to find a new collision, where l denotes the encoded bit length of the message. For preimage attack on JH-s, after a $\frac{1}{e}2^{(s+l)/2+n}$ precomputing, the adversary needs $2^{(s+l)/2}$ queries to the underlying compression function to find a new preimage. If $s = 224$, the attacker only needs 2^{56} and 2^{112} compression function queries to mount a new collision attack and preimage attack on JH-224 without length padding respectively. Table. 1 lists the attack complexity of our attack on JH variants.

The attack is easily extended to JH's variant MJH. For the $2n$-bit MJH, after a precomputing, the adversary needs about $2^{n/2}$ queries to the blockcipher to find a new collision and 2^n queries to the blockcipher to find a new (2nd) preimage.

In the structure of Grøstl-s hash function, two permutation F and Q are used, see Fig. 6. For collision attack on Grøstl-s without length padding, we needs about $2^{s/2}$ queries to F and $2^{s/2}$ queries to Q to find a collision with probability 0.39. The best known collision attack requires $3 \times 2^{s/2}$ queries to F and $2^{s/2+1}$ queries to Q. For (second) preimage attack on Grøstl-s without length padding, this attack requires about 2^s queries to F and 2^s queries to Q, where the best known (second) preimage attack requires 3×2^s queries to F and 2^{s+1} queires to Q. Thus for Grøstl, the query complexity of our collision and preimage attack are one half of birthday collision attack and exhaustive preimage attack respectively.

Our attacks exploit structure flaws in the design of JH, Grøstl and SMASH. The results show that these constructions are weaker than Sponge construction. At the same time the provable security of chopMD in the literature is challenged. Through our attack, it is easy to see that the chopMD mode used in JH or Grøstl don't improve its security.

2 Preliminaries

General Notation. For two bitsrings x and y, $x \parallel y$ denotes the concatenation of x and y. A blockcipher E with n-bit block and n-bit keysize is called an (n, n) blockcipher.

Information Theoretic Model. In the information theoretic model, the adversary is computationally unbounded but is given up to q queries to the underlying ideal primitive. The advantage of the adversary is related to the query times q. Almost every security proof in the hash function literature uses this model.

The ChopMD Mode. The chopMD is an iteration mode same as the plain Merkle-Damgärd mode except that the final output is truncated. The first formal

security proof of chopMD [5] is by Coron *et al.* This mode is adopted by SHA-3 winner Keccak [3] and the other two finalists JH and Grøstl.

Even-Mansour Structure. In [7], Even and Mansour proposed a scheme for a block cipher which uses a fixed n-bit permutation F. The n-bit plaintext is first xored with n-bit K_1, then the result is the input of the permutation, the output of the permutation is next xored with n-bit K_2, and the result is the ciphertext. In [6], Dunkelman and Shamir show that the original two-key Even-Mansour structure is not minimal since it can be simplified into a single key structure with $K_1 = K_2 = K$. The two-key and single-key Even-Mansour structure are shown in Fig. 1.

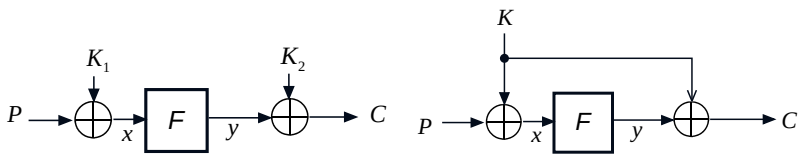

Fig. 1. Two-Key Even-Mansour Structure and Single-Key Even-Mansour Structure

3 The Even-Mansour structure Hash Functions

In this section we introduce the high level structure of JH, Grøstl and SMASH. Actually they are based on the Even-Mansour structure where the message is the key and the chain value is the plaintext/ciphertext of the Even-Mansour structure.

Let F be a $2n$ bit permutation, L_1, L_2 be two transformation on $2n$ bits. The high level compression function of JH, Grøstl and SMASH can be denoted as

$$h_i = F(h_{i-1} \oplus m_i) \oplus L_1(m_i) \oplus L_2(h_{i-1})$$

where $h_{i-1}, h_i, m_i \in \{0,1\}^{2n}$. Let $chop_s$ be the last s bits of a $2n$-bit value and h_0 be a fixed initial value IV. We call this high level structure Even-Mansour hash structure. The 2-block Even-Mansour hash structure with chopMD mode of operation is depicted in Fig.2.

In the next we give attacks on Even-Mansour hash structure with chopMD mode of operation. The attack can be easily applied to the JH, Grøstl, SMASH and their variants MJH, SMASH *et al.*

3.1 Collision Attack on Even-Mansour chopMD Hash

We assume F be an ideal permutation and the adversary never makes repeat queries. That is to say, the adversary never makes queries that she already knows

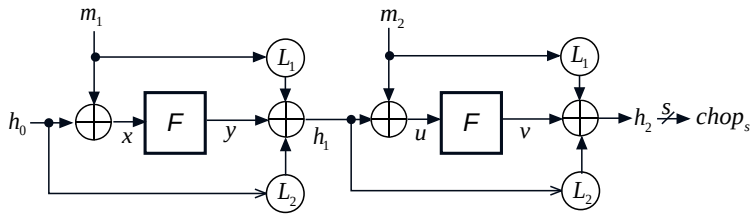

Fig. 2. The 2-block Even-Mansour hash with chopMD mode.

the result. In the attack we use two blocks of message (m_1, m_2) and fix the initial value as $h_0 = IV$ where IV is a $2n$-bit constant. As shown in Fig.2, we let (x, y) be the input-output queried pairs in the first block query and (u, v) be the input-output queried pairs in the second block query. Thus we have

$$\begin{aligned} m_1 &= h_0 \oplus x \\ h_1 &= L_1(m_1) \oplus L_2(h_0) \oplus y \\ m_2 &= h_1 \oplus u \\ h_2 &= L_1(m_2) \oplus L_2(h_1) \oplus v. \end{aligned}$$

The attack is described as follows:

1. Set h_0 to be the constant IV.
2. Choose r random values of x and make queries to F, we thus get r random values of (x, y). Since $h_1 = y \oplus L_1(m_1) \oplus L_2(h_0)$, we get r random values of h_1.
3. Choose r random values of u and make queries to F, we thus get r random values of (u, v).
4. For each value of (u, v), we can compute $m_2 = h_1 \oplus u$ and $h_2 = v \oplus L_1(m_2) \oplus L_2(h_1)$. Since there are r random values of h_1, we can get r random values of m_2 and h_2.
5. There are total r values of (u, v), thus we can get r^2 random values of h_2.
6. If the final hash value is truncated to s bits where $s \leq 2n$. Let r be $2^{s/4}$, thus we can get $r^2 = 2^{s/2}$ random values of h_2 and $chop_s$. According to the birthday paradox, there exists two pairs of (h_1, m_2) colliding at $chop_s(h_2)$ with probability 0.39.
7. The adversary needs $2 \times 2^{s/4} = 2^{s/4+1}$ queries to the permutation F to find a collision with probability 0.39.
8. If the message length is encoded into l bits and appended into the second message block m_2, there are $r^2/2^l$ values of m_2 have the same last l bits. Let $r^2/2^l = 2^{s/2}$, we have $r = 2^{s/4+l/2}$. The adversary needs $2^{s/4+l/2+1}$ queries to the underlying compression function to find a collision with probability 0.39.

3.2 Preimage and Second Preimage Attack

For preimage and second preimage attack, we need to find a preimage for a s-bit value. It is easy to see that if we let $r = 2^{s/2}$, at last we can get $r^2 = 2^s$ random values of $chop_s$, since $chop_s$ is s bits, with high probability we can find a (second) perimage. Thus the adversary needs $2 \times 2^{s/2} = 2^{s/2+1}$ queries to the permutation F to find a preimage with high probability.

If the message length is encoded into l bits and appended into the second message block m_2, there are $r^2/2^l$ values of m_2 have the same last l bits. Let $r^2/2^l = 2^s$, we have $r = 2^{(s+l)/2}$. The adversary needs $2^{(s+l)/2+1}$ queries to the underlying compression function to find a preimage with probability close to 1.

4 The JH hash function

The compression function of JH is a special case of Even-Mansour hash where the transformation $L_1(m \parallel 0^n) = (0^n \parallel m)$ and L_2 is a zero transformation.

Let F be a $2n$ bit permutation, the compression function of JH depicted in Fig. 3 is defined as:

$$f(h_{i-1}, g_{i-1}, m_i) = F(h_{i-1} \oplus m_i \parallel g_{i-1}) \oplus (0^n \parallel m_i)$$

where $h_{i-1}, g_{i-1}, m_i \in \{0,1\}^n$.

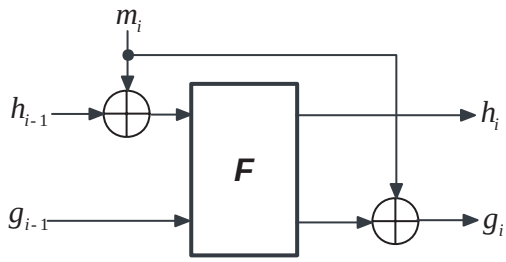

Fig. 3. The compression function of JH.

JH uses the chopMD mode of operation. The initial value is fixed as (IV_0, IV_1) and the message M is first padded into l message blocks, then the usual Merkle-Damgård iteration is applied to F to compute the last chain value (h_l, g_l). The final output is is the last s bits of g_l.

5 Attacks on JH and MJH Hash Functions

5.1 Attacks on 3-block JH

The 3-block JH-s hash function is shown in Fig.4. Let (h_0, g_0) be a fixed initial value (IV_0, IV_1). From the figure, we have

$$(h_1, g_1) = F(h_0 \oplus m_1 \parallel g_0) \oplus (0^n \parallel m_1)$$
$$(h_2, g_2) = F(h_1 \oplus m_2 \parallel g_1) \oplus (0^n \parallel m_2)$$
$$(h_3, g_3) = F(h_2 \oplus m_3 \parallel g_2) \oplus (0^n \parallel m_3).$$

As in the figure, the output of a query (x, g_0) to the first block F is denoted as (h_1, y), and the output of a query (u, g_2) to the third block F is denoted as (h_3, v). The final output of JH-s is the last s bits of g_3 and denoted as $chop_s$.

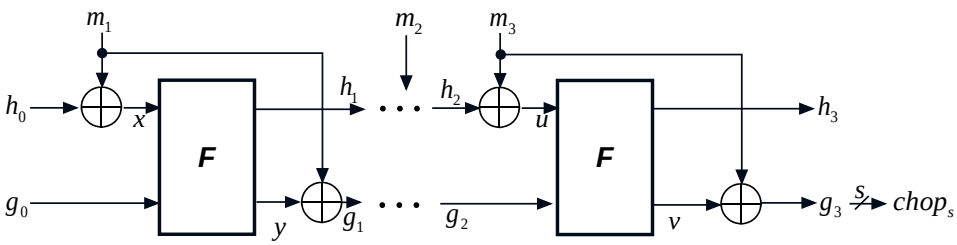

Fig. 4. The 3-block structure of JH-s. All wires carry n-bit values.

The collision and (2nd) preimage attack is described as follows:

– Collision attack:
  1. $h_0 \parallel g_0$ is fixed to the initial value.
  2. *Precomputing*: Choose many random values of $m_1 \parallel m_2$ and make queries to F until a r-collision at g_2 is found. At last we get r random values $(h_2^i \parallel g_2), 1 \le i \le r$.
  3. Choose r random values $u^i, 1 \le i \le r$, make queries $(u^i \parallel g_2), 1 \le i \le r$ to F, we thus get r random values $h_3^i \parallel v^i), 1 \le i \le r$.
  4. For each value of $(u^i \parallel g_2, h_3^i \parallel v^i), 1 \le i \le r$, we can compute $m_3 = h_2 \oplus u^i$ and $g_3 = v^i \oplus m_3$. Since there are r random values of h_2, we can get r random values of m_3 and g_3.
  5. There are total r values of $(u^i \parallel g_2^i, h_3^i \parallel v^i)$, thus we can get r^2 random values of g_3.
  6. If the final hash value is truncated to s bits where $s \le n$. Let r be $2^{s/4}$, thus we can get $r^2 = 2^{s/2}$ random values of h_3 and $chop_s$. According to the birthday paradox, there exists two pairs colliding at $chop_s$ with probability 0.39.

– Preimage attack: For preimage and second preimage attack, we need to find a preimage for a s-bit value. It is easy to see that if we let $r = 2^{s/2}$, we can get $r^2 = 2^s$ random values of $chop_s$, since $chop_s(h_2)$ is only s bits, with probability close to 1 we can find a (second) perimage.

5.2 Attack Complexity.

	Collision Attack Complexity			Preimage Attack Complexity		
Variants	*Precomputing*	*Attack*	*Best Known*	*Precomputing*	*Attack*	*Best Known*
JH-s	$2^{s/4+n}$	$2^{s/4}$	$2^{s/2}$	$2^{s/2+n}$	$2^{s/2}$	2^s
JH$'$-224	2^{568}	2^{120}	2^{112}	2^{624}	2^{176}	2^{224}
JH$'$-256	2^{576}	2^{128}	2^{128}	2^{640}	2^{192}	2^{256}
JH$'$-384	2^{608}	2^{160}	2^{192}	2^{704}	2^{256}	2^{384}
JH$'$-512	2^{640}	2^{192}	2^{256}	2^{768}	2^{320}	2^{512}

Table 1. The query complexity of collision attack and preimage attack on JH variants. JH-s is the s-bit JH hash function without length padding to the last block. JH$'$-s variants mean the length is appended to the message and padded to the last block.

In the collision attack, the precomputing steps cost $T_{r-collision}$ time to find a r-collision. After that, the adversary needs $2^{s/4}$ queries to the permutation F to find a collision with probability 0.39.
In [21], Suzuki *et al.* analyzed the concrete probability of a r-collision. They showed that by querying $(r!)^{1/r} \times 2^{n(r-1)/r}$ times, a r-collision is found with probability approximately 0.5. By using Stirling's approximation, we have

$$r! \approx (r/e)^r \sqrt{2\pi r},$$

thus $T_{r-collision} \approx (r!)^{1/r} \times 2^{n(r-1)/r} \approx \frac{r}{e} 2^n$. If $r = 2^{s/4}$, $T_{r-collision}$ is about $2^{s/4+n}/e$.
If the message length is encoded into l bits and appended into the third message block m_2, there are $r^2/2^l$ values of m_2 have the same last l bits. Let $r^2/2^l = 2^{s/2}$, then $r = 2^{s/4+l/2}$.

1. Thus for JH-512, $l = 128$, the adversary needs $2 \times 2^{512/4+128/2} = 2^{192}$ queries to F to find a collision with probability 0.39.
2. For JH-384, the adversary needs $2 \times 2^{384/4+128/2+1} = 2^{160}$ queries to F to find a collision with probability 0.39.

3. For JH-256, the adversary needs $2 \times 2^{256/4+128/2+1} = 2^{128}$ queries to F to find a collision with probability 0.39.
4. For JH-224, the adversary needs $2 \times 2^{224/4+128/2+1} = 2^{120}$ queries to F to find a collision with probability 0.39.

In the preimage attack, the precomputing steps cost $T_{r-collision}$ time to find a r-collision. After that, the adversary needs $2^{s/2}$ queries to the permutation F to find a (2nd) preimage with probability close to 1.

If the message length is encoded into l bits and appended into the second message block m_2, there are $r^2/2^l$ values of m_2 have the same last l bits. Let $r^2/2^l = 2^s$, we have $r = 2^{(s+l)/2}$. The adversary needs $2^{(s+l)/2}$ queries to the underlying compression function to find a preimage with probability close to 1.

1. Thus for JH-512, $l = 128$, the adversary needs $2^{(512+128)/2} = 2^{320}$ queries to F to find a preimage with probability close to 1.
2. For JH-384, the adversary needs $2^{(384+128)/2} = 2^{256}$ queries to F to find a preimage with probability close to 1.
3. For JH-256, the adversary needs $2^{(256+128)/2} = 2^{192}$ queries to F to find a preimage with probability close to 1.
4. For JH-224, the adversary needs $2^{(224+128)/2} = 2^{176}$ queries to F to find a preimage with probability close to 1.

The attack complexity is shown in Table 1. Note that the original JH hash function uses an extra message block loadding the message length, our attacks use variants padding of JH where the length is appended to the message and then padded to the third message block.

Our attack exploits weakness of JH hash function. To mount collision or (second) preimage attacks, the adversary first find a multi-collision with precomputing, then he can find a new collision or preimage in very low complexity.

5.3 Attacks on JH's variant MJH

MJH hash function is proposed by Lee and Stam [16]. It is a variant of JH hash function. It uses two calls to a (n, n)-bit blockcipher E to implement the underlying primitive F, while F needn't to be a permutation. Let σ be an involution on $\{0,1\}^n$ with no fixed point, and let $\theta \neq 0, 1$ be a constant in $\mathbb{F}_{2^n}$, the primitive F is defined as

$$\begin{aligned} F[\sigma,\theta] : \{0,1\}^{2n} &\longrightarrow \{0,1\}^{2n} \\ (x_L \parallel x_R) &\longrightarrow (y_L \parallel y_R) \\ y_L &= E_{x_R}(x_L) \oplus x_L \\ y_R &= \theta \cdot (E_{x_R}(\sigma(x_L)) \oplus \sigma(x_L)) \oplus x_L. \end{aligned}$$

By applying the JH transform, the compression function of MJH is the same as in Fig. 3. Then MJH uses Merkle-Damgård mode without length padding to the last block to calculate the final $2n$-bit hash value.

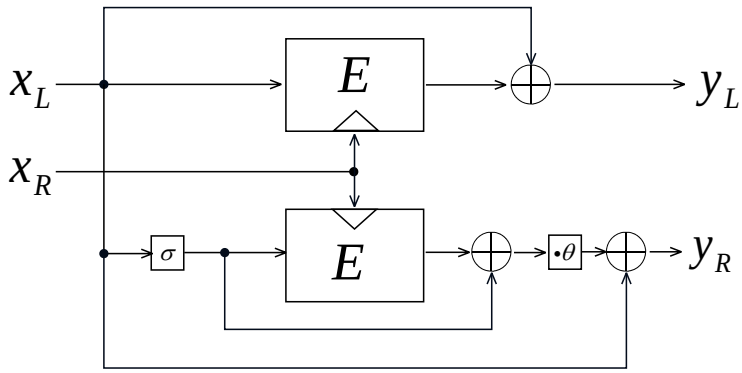

Fig. 5. The F primitive in MJH, where $(y_L, y_R) = F(x_L, x_R)$. All wires carry n-bit values. E is an (n, n) blockcipher. σ is an involution and θ is a constant in $\mathbb{F}_{2^n} \setminus \mathbb{F}_2$.

Due to the involution property, the adversary can get a pair of $(x_L \parallel x_R, y_L \parallel y_R)$ when she makes a query $E_{x_R}(x_L)$ to the upper blockcipher and a query $E_{x_R}(\sigma(x_L))$ to the lower blockcipher. That is to say, for each query $x_L \parallel x_R$ to the primitive F, the adversary can get two pairs of $(x_L \parallel x_R, y_L \parallel y_R)$ by making two blockcipher queries.

Since MJH outputs $2n$ bits as the hash value, thus the collision attack and (2nd) preimage attack is a little different from the attack on JH. The attack is similar as in Fig. 4 and described as follows.

– Collision attack:
  1. $h_0 \parallel g_0$ is fixed to the initial value.
  2. *Precomputing*: Choose many random values of $m_1 \parallel m_2$ and make queries to F until a r-collision at g_2 is found where $r = 2^{n/2-1}$. At last we can get $2r$ random values $(h_2^i \parallel g_2), 1 \le i \le r$.
  3. Choose $2r$ random values $u^i, 1 \le i \le 2r$, make queries $(u^i \parallel g_2), 1 \le i \le 2r$ to F, we thus get $4r$ random values $h_3^i \parallel v^i, 1 \le i \le 4r$ by $2r$ blockcipher queries. Due to birthday paradox, with probability $1 - e^{-\frac{(2^{n/2+1})^2}{2\times 2^n}} \approx 0.86$ we can obtain a pair $(u^i \parallel g_2, u^j \parallel g_2), 1 \le i < j \le 2r$ colliding at h_3.
  4. For the value $u^i \parallel g_2$, we can compute $m_3 = h_2 \oplus u^i$ and $g_3 = v^i \oplus m_3$. Since there are $2^{n/2}$ random values of h_2, we can get $2^{n/2}$ random values of m_3 and g_3. For the value $u^j \parallel g_2$, we can also get $2^{n/2}$ random values of m_3 and g_3. Thus with probability $1 - e^{-\frac{1}{2}} \approx 0.39$ a match will be found for these two sets.
  5. The adversary needs $2^{1.5n}$ queries in pre-computing, then he needs $2^{n/2}$ queries to the blockcipher to find a collision with probability $0.86 \times 0.39 \approx 0.34$.

– Preimage attack: For preimage and second preimage attack, we need to find a preimage for a $2n$-bit value. It is easy to see that if we let $r = 2^{n/2-1}$, after find a hitting at h_3, we can get 2^n random values of g_3. Since g_3 is only n bits, with probability close to 1 we can find a (second) perimage. Thus the adversary needs 2^{3n} queries in pre-computing, then he needs 2^n queries to the blockcipher to find a (2nd) preimage with probability close to 1.

6 The Grøstl hash function

The compression function of Grøstl is a special case of Even-Mansour hash where the transformation L_1 is the Q permutation defined in [10] and L_2 is an identity transformation.

Let F and Q be a $2n$ bit permutation, the compression function of Grøstl depicted in Fig. 6 is defined as:

$$f(h_{i-1}, m_i) = F(h_{i-1} \oplus m_i) \oplus Q(m_i) \oplus h_{i-1}$$

where $h_{i-1}, m_i \in \{0,1\}^{2n}$.

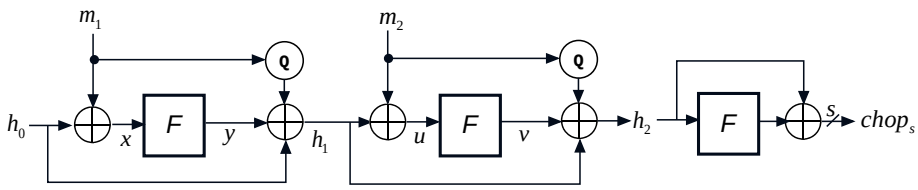

Fig. 6. The two-block of Grøstl.

Grøstl uses the chopMD mode of operation. The initial value is fixed as IV and the message M is first padded into l message blocks, then the usual Merkle-Damgård iteration is applied to F to compute the last chain value h_l. At last a final transformation $\Omega(x) = F(x) \oplus x$ is applied to h_l, the final output is is the last s bits of $\Omega(h_l)$.

6.1 Collision attack on Grøstl without length padding

We assume F and Q be two ideal permutations and the adversary never makes repeat queries. That is to say, the adversary never makes queries that she already knows the result. In the attack we use two blocks of message (m_1, m_2) and fix the initial value as $h_0 = IV$ where IV is a $2n$-bit constant. As shown in Fig.6, we let (x, y) be the input-output queried pairs in the first block query and (u, v)

be the input-output queried pairs in the second block query. Thus we have

$$\begin{aligned} m_1 &= h_0 \oplus x \\ h_1 &= Q(m_1) \oplus h_0 \oplus y \\ m_2 &= h_1 \oplus u \\ h_2 &= Q(m_2) \oplus h_1 \oplus v. \end{aligned}$$

The attack is described as follows:

1. Set h_0 to be the constant IV.
2. Choose r random values of x and make queries to F, we thus get r random values of (x, y). Since $h_1 = y \oplus Q(m_1) \oplus h_0$, we get r random values of h_1.
3. Choose r random values of u and make queries to F, we thus get r random pairs of (u, v).
4. For each value of (u, v), we can compute $m_2 = h_1 \oplus u$ and $h_2 = v \oplus Q(m_2) \oplus h_1$. Since there are r random values of h_1, we can get r random values of m_2 and h_2.
5. There are total r values of (u, v), thus we can get r^2 random values of h_2.
6. If the final hash value is truncated to s bits where $s \leq n$. Let r be $2^{s/4}$, thus we can get $r^2 = 2^{s/2}$ random values of h_2 and $chop_s$. According to the birthday paradox, there exists two pairs of (h_1, m_2) colliding at $chop_s(h_2)$ with probability 0.39.
7. The adversary needs $2 \times 2^{s/4} + 2^{s/2} = 2^{s/4+1} + 2^{s/2}$ queries to the permutation F and $2^{s/4} + 2^{s/2}$ queries to permutation Q to find a collision with probability 0.39. The best known collision attack requires $3 \times 2^{s/2}$ queries to F and $2^{s/2+1}$ queries to Q.

6.2 (Second) preimage attack on Grøstl without length padding

For preimage and second preimage attack, we need to find a preimage for a s-bit value. It is easy to see that if we let $r = 2^{s/2}$, at last we can get $r^2 = 2^s$ random values of $chop_s$, since $chop_s$ is s bits, with high probability we can find a (second) perimage. This attack requires $2^{s/2+1} + 2^s$ queries to F and 2^s queries to Q, where the best known (second) preimage attack requires 3×2^s queries to F and 2^{s+1} queires to Q.

Our attacks exploit weakness of Grøstl construction. The design structure of Grøstl requires more computation than SHA-3 winner Keccak, but has less security against our attack.

7 The SMASH Hash Function and its variants

The SMASH hash function is proposed by Knudsen [12]. Its structure is the same as Even-Mansour hash depicted in Fig. 2 except the L_1 transformation is defined as $L_1(x) = x \cdot \theta$ where $\cdot$ is the multiply operation in the Galois Field $\mathbf{GF}(2^{2n})$ and θ is an arbitrary field element in $\mathbf{GF}(2^{2n})$ with restriction $\theta \neq 0, 1$.

After the proposal, Pramstaller *et al.* gave collision attacks and Lamerger *et al.* gave second preimage attacks on it [19,13,14]. Knudsen suggested two tweaked versions of SMASH to thwart the attack. Later Fouque *et al.* proposed attacks on the two tweaked versions and proposed a generalization of SMASH using two permutations [9]. This generalization is proved secure against collision and preimage attack in the ideal permutation model in $\Omega(2^n)$ queries and $\Omega(2^{n/2})$ respectively, where the hash value is $2n$ bits. Fouque *et al.* also proposed a non-trival collision attack in $\Omega(2^{3n/8})$ queries. There is no collision attack on this generlization in $\Omega(2^{n/2})$ queries in the current literature. Note this generlization is just the Even-Mansour structure where L_1 and L_2 are two permutations.

It is easy to see our attack is directly applied to SMASH and its generalization suggested by Fouque *et al.* The complexity is the same as the attack on Even-Mansour hash, thus we omit the details here.

8 Conclusion

In this paper we have presented collision and preimage attacks on Even-Mansour hash functions JH, MJH, Grøstl, SMASH and their variants. Our attacks exploit structure flaws in the design of these hash functions. Our attacks invalidate some previous security proofs for these hash functions.

Even-Mansour hash functions are weaker than Sponge since the message block (with or without a transformation) is both added to the input and the output of the underlying permutation.

Through our anlysis, the chopMD design philosophy is also challenged since it does not improve the security for Even-Mansour hash functions. That is, if there exists some special weakness in the compression function just like in the analysis of JH, Grøstl and SMASH, after applying to chopMD mode, the security of the hash function against collision and preimage attack is not improved.

We also realized our attack can also be applied to other popular blockcipher-based hash functions, such as MD4-family hash functions, including MD4,MD5 and SHA-1. The following research focus on generic attacks on blockcipher-based hash functions.

References

1. Elena Andreeva, Bart Mennink, and Bart Preneel. On the indifferentiability of the grøstl hash function. In J.A. Garay Prisco and R. De, editors, *SCN 2010*, volume LNCS 6280, pages 88–105. Springer-Verlag, 2010.
2. G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. On the indifferentiability of the sponge construction. In *Advances in Cryptology - EUROCRYPT'08*, volume LNCS 4965, pages 181–197, Istanbul, Turkey, 2008. Springer-Verlag.
3. G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. The Keccak reference. Submission to NIST, 2011.
4. Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. Security analysis of the mode of JH hash function. In *FSE 2010*, volume LNCS 6147, pages 168–191. Springer-Verlag, 2010.

5. J. S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-damgård revisited: How to construct a hash function. In *Advances in Cryptology - CRYPTO'05*, volume LNCS 3621, pages 430–448. Springer-Verlag, 2005.
6. O. Dunkelman, N. Keller, and A. Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In D. Pointcheval Johansson and T., editors, *EUROCRYPT 2012*, volume LNCS 7237, pages 336–354. Springer-Verlag, 2012.
7. S. Even and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of Cryptology*, 10(3):151–162, 1997.
8. FIPS. FIPS 180-1 Secure Hash Standard. Federal Information Processing Standard (FIPS), Publication 180-1, National Institute of Standards and Technology, US Department of Commerce, Washington D.C, 1995.
9. Pierre-Alain Fouque, Jacques Stern, and Sébastien Zimmer. Cryptanlysis of tweaked versions of SMASH and reparation. In R. Avanzi Sica, L. Keliher, and F., editors, *SAC 2008*, volume LNCS 5381, pages 136–150. Springer-Verlag, 2009.
10. P. Gauravaram, L. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S. Thomsen. Grøstl - a SHA-3 candidate (2009).
11. Deukjo Hong and Kwon. Cryptanalysis of some double-block-length hash modes of block ciphers with n-bit block and n-bit key. http://eprint.iacr.org/2013/174, 2013.
12. L. R. Knudsen. SMASH - a cryptographic hash function. In H. Gilbert Handschuh and H., editors, *FSE 2005*, volume LNCS 3557, pages 228–242. Springer-Verlag, 2005.
13. M. Lamberger, N. Pramstaller, C. Rechberger, and V. Rijmen. Second preimages for SMASH. In M. Abe, editor, *CT-RSA 2007*, volume LNCS 4377, pages 101–111. Springer-Verlag, 2007.
14. Mario Lamberger, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of the hash function design strategy called smash. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 54(8):3647–3655, 2008.
15. Jooyoung Lee and Deukjo Hong. Collision resistance of the JH hash function. *IEEE Transaction on Information Theory*, 58(3):1992–1995, 2012.
16. Jooyoung Lee and Martijn Stam. MJH: A faster alternative to MDC-2. In *CT-RSA 2011*, volume LNCS 6558, pages 213–236. Springer-Verlag, 2011.
17. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
18. Dustin Moody, Souradyuti Paul, and Daniel Smith-Tone. Improved indifferentiability security bound for the jh mode. In *http://eprint.iacr.org/2012/278.pdf*.
19. N. Pramstaller, C. Rechberger, and V. Rijmen. Breaking a new hash function design strategy called SMASH. In B. Preneel Tavares and S., editors, *SAC 2005*, volume LNCS 3897, pages 233–244. Springer-Verlag, 2005.
20. R. L. Rivest. The MD5 message digest algorithm. In *Request for Comments (RFC) 1321*. Internet Activities Board, Internet Privacy Task Force, 1992.
21. K. Suzuki, D. Tonien, K. Kurosawa, and K. Toyota. Birthday paradox for multi-collisions. In M. S. Rhee Lee and B., editors, *ICISC 2006*, volume LNCS 4296, pages 29–40. Springer-Verlag.
22. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRPTO'05*, volume LNCS 3621, pages 17–36, Santa Barbara, CA, USA, 2005. Springer-Verlag.
23. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT'05*, volume LNCS 3494, pages 19–35, Aarhus, Denmark, 2005. Springer-Verlag.

24. Hongjun Wu. The hash function JH. http://www3.ntu.edu.sg/home/wuhj/research/jh/jh-round3.pdf, 2011.