Functional Graph Revisited: Updates on (Second) Preimage Attacks on Hash Combiners

Zhenzhen Bao[1,2], Lei Wang[1,3,4⋆], Jian Guo[2], and Dawu Gu[1]

[1] Shanghai Jiao Tong University, Shanghai, China
[2] Nanyang Technological University, Singapore
[3] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[4] Westone Cryptologic Research Center, Beijing, China
baozhenzhen10@gmail.com, {wanglei_hb,dwgu}@sjtu.edu.cn, guojian@ntu.edu.sg

Abstract. In this paper, we study functional-graph-based (second) preimage attacks against hash combiners. Our contributions are threefold:

– in EUROCRYPT 2016, Dinur proposes generic (second) preimage attacks on the concatenation combiner and the XOR combiner using a new and essential observation on functional graph, which is experimentally verified but the proof is incomplete. Our first contribution is to provide a proof for Dinur's observation;
– we find improved preimage attack against the XOR combiner with a complexity of $2^{5n/8}$, while the previous best-known complexity is $2^{2n/3}$;
– we find the first generic second-preimage attack on Zipper hash with an optimal complexity of $2^{3n/5}$.

Keywords: Hash Combiner · Functional Graph · XOR Combiner · Zipper Hash · (Second) Preimage Attack

1 Introduction

A cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$ maps arbitrarily long messages to n-bit digests. It is a fundamental primitive in modern cryptography and has been widely utilized in various cryptosystems. There are three *basic* security requirements on a hash function $\mathcal{H}$:

- **Collision Resistance.** It must be computationally infeasible to find two distinct messages M and M' such that $\mathcal{H}(M) = \mathcal{H}(M')$;
- **Second Preimage Resistance.** Given a message M, it must be computationally infeasible to find a message M' such that $M' \neq M$ and $\mathcal{H}(M') = \mathcal{H}(M)$;

⋆ Corresponding author

- **Preimage Resistance.** Given a target hash digest V, it must be computationally infeasible to find a message M such that $\mathcal{H}(M) = V$.

As generic birthday attack and the brute-force attack require complexities of $2^{n/2}$ and 2^n to find a collision and a (second) preimage, respectively. It is expected that a secure hash function should provide the same security level of resistance.

Among various approaches of designing a hash function, one is to build a hash combiner from two (or more) hash functions in order to achieve security amplification, that is the hash combiner has higher bound of security resistance than its underlying hash functions, or to achieve security robustness, that is the hash combiner is secure as long as (at least) any one of its underlying hash functions is secure. In particular, hash combiners were used in practice, e.g., in SSL [14] and TLS [2].

Concatenation combiner and XOR combiner are the two most classical hash combiners. Using two (independent) hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$, the former concatenates their outputs: $\mathcal{H}_1(M)\|\mathcal{H}_2(M)$, and the latter XORs their outputs: $\mathcal{H}_1(M) \oplus \mathcal{H}_2(M)$. From a theoretical point of view, the concatenation combiner is robust with respect to collision resistance, and the XOR combiner is robust with respect to PRF and MAC in the black-box reduction model [22]. Advanced security amplification combiners and robust multi-property combiners for hash functions have been constructed [9,10,11,12]. More generally[5], cryptographers have also studied cascade constructions of two (or more) hash functions, that is to compute $\mathcal{H}_1$ and $\mathcal{H}_2$ in a sequential order. Well-known examples are Hash Twice: $\mathcal{H}_2(\mathcal{H}_1(IV, M), M)$ and Zipper Hash [25]: $\mathcal{H}_2(\mathcal{H}_1(IV, M), \overleftarrow{M})$, where $\overleftarrow{M}$ is the reversed (block) order of original message M. We regard these cascade constructions of hash functions as hash combiners in this paper.

This paper is mainly interested in combiners of *iterative* hash functions, in particular following the Merkle-Damgård construction [27,6]. An iterative hash function splits a message M into blocks $m_1, \ldots, m_\ell$ of fixed length, and processes these blocks by iterating a compression function h (or a series of compression functions) over an internal state x with an initial constant denoted as IV. Finally, the hash digest is computed by a finalization function with the bit length of M denoted as $|M|$ as input. The finalization function can be either the compression function h or another independent function. For the simplicity of description, we fix the finalization function as h in the rest of the paper, but we stress that our attacks also work in a straight forward way for the case of an independent finalization function. We mainly focus on narrow-pipe iterative hash functions, *i.e.*, every internal state x_i $(0 \le i \le \ell)$ have the same bit length with the output hash digest.

$$x_0 = IV \qquad\qquad x_{i+1} = h(x_i, m_{i+1}) \qquad\qquad \mathcal{H}(M) = h(x_\ell, |M|)$$

[5] Here we need to generalize the syntax of hash functions such that the initial value is also regarded as an input parameter.

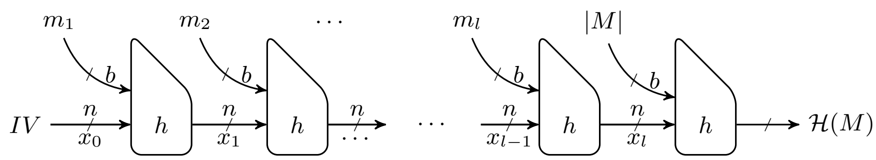

Fig. 1: Narrow-pipe Merkle-Damgård Hash Function

Combiners of iterative hash functions have received extensive analysis. Several generic attacks have been devised on the above combiners, which can work even with *ideal* compression functions, indicating the upper security bound of these combiners. In a seminal paper [20], Joux presents a technique to find multicollision on an iterative hash function that has a complexity not much greater than that of finding a single collision. Based on this technique, he finds collision and preimage attacks on the concatenation combiner with complexities much lower than expected, and shows that it offers essentially the same security level with a single n-bit hash function.[6] In [24], Leurent and Wang propose an interchange structure that can break the pairwise dependency between the internal states of two iterative hash functions during computing a common message. Based on this structure, they are able to compute the two hash functions independently, and then launch a meet-in-the-middle preimage attack on the XOR combiner with a complexity exponentially lower than 2^n, more precisely $\tilde{\mathcal{O}}(2^{5n/6})$.

For combiners of cascade constructions, towards the basic security requirements, a second preimage attack on Hash twice has been published by Andreeva *et al.* in [3] with a complexity of $\mathcal{O}(2^n/L)$, where L is the block length of the challenging message. On the other hand, there is no generic attack on Zipper hash with respect to the basic security notions, which is highlighted as an *open problem* in [24]. Besides, cryptographers have also analyzed the resistance of Hash twice and Zipper hash with respect to other security notions such as multi-collision, herding attack, etc. Examples include [28,17,3,19].

Very recently Dinur in [7] publishes new generic attacks on the concatenation combiner and the XOR combiner. He finds a second preimage attack on the concatenation combiner with a complexity of optimally $2^{3n/4}$, and an improved preimage attack on the XOR combiner with a complexity of optimally $2^{2n/3}$. Differently from previous attacks on combiners in [20,24] which are mainly based on collision-finding techniques [20,21], one main technical contribution of Dinur's attacks is to exploit properties of functional graph of a random mapping. More specifically, one can fix the message input as a constant, and then turn the compression function h to an n-bit to n-bit random mapping. It has many interesting properties, and has been extensively studied and utilized in cryptography. Examples include [16,29,31,23,32,8,15,30]. Besides using known functional graph

[6] In fact, Joux's attacks require that only one hash function is iterative and narrow-pipe.

properties, Dinur finds an observation, which is essential for the complexity advantage of his attacks on those combiners. The observation is (briefly) described as follows. For two (independent) n-bit functional graphs defined by f_1 and f_2, let $\bar{x}$ and $\bar{y}$ be two iterates of depth 2^t in f_1 and f_2 respectively, that is $\bar{x}$ and $\bar{y}$ are images of 2^t iterations on f_1 and f_2 respectively. For a pair of random nodes x_r and y_r in the functional graphs defined by f_1 and f_2 respectively, compute a chain by iteratively applying f_1 and f_2 to update x_r and y_r until a maximum length of 2^t.

Dinur's observation [7, Sec. 3.3]. *The probability of x_r and y_r being iteratively updated to $\bar{x}$ and $\bar{y}$ at a common distance is 2^{3t-2n}.*

Although experimentally verified, the proof of this observation is incomplete in [7]. With the number of chains increasing, there will be dependency between chains due to colliding and merging, which cannot be analyzed probabilistically anymore. We refer to [7] for more details. Thus, this observation stays as rather a conjecture.

Lines of research of combining iterative hash functions also include the study of hash combiners with weak compression functions, *i.e.*, the attacker is given additional interfaces to receive random preimages of the compression functions [25,18,5,19], and analysis of combiners of dedicated hash functions [26]. In particular, the concatenation combiner, the XOR combiner and Zipper hash with weak compression functions have been proven in [25,18] to be indifferentiable from a random oracle with an $n/2$-bit security, indicating the lower security bound regarding basic security notions for these combiners.

1.1 Our Contributions

This paper investigates functional graph of a random mapping, and based on its properties evaluates the security of hash combiners. We have mainly three contributions. Firstly, we provide a rigorous proof for Dinur's observation described above, which makes it *solid* for future usage in cryptography and other research fields. Instead of the approach of viewing chains as random chains and studying them probabilistically as the informal discussions in [7], we derive and utilize certain *statistical* properties of functional graph of a random mapping, *i.e.*, the number of k-th iterate image points, to complete the proof.

Secondly, we find an improved preimage attack on the XOR combiner, by exploiting the cyclic nodes in a functional graph. One main step in previous preimage attack on the XOR combiner is to search for a pair of nodes, x in functional graph of a function f_1 and y in functional graph of another function f_2, which reach to a pair of predefined nodes $\bar{x}$ of f_1 and $\bar{y}$ of f_2 at a common distance. We find that the probability of a random pair x_r and y_r reaching to $\bar{x}$ and $\bar{y}$ at a common distance can be greatly amplified, by exploiting some property of cyclic nodes as follows. When applying a function f to update a cyclic node in its functional graph iteratively, the cyclic node loops along the cycle and goes back to itself after a number of multi-cycle-length function calls.

This property of cyclic nodes turns out to be very beneficial for finding a pair (x, y) that reach to $(\bar{x}, \bar{y})$ at a common distance. More specifically, $\bar{x}$ and $\bar{y}$ are predefined to be cyclic nodes within the largest components in the functional graphs of f_1 and f_2 respectively. Suppose a random pair of x_r and y_r reach to $\bar{x}$ and $\bar{y}$ at distances of d_1 and d_2 respectively. We can try correcting the distance bias $d_1 - d_2 \neq 0$, by letting $\bar{x}$ and $\bar{y}$ loop along their cycles. Note these two cycles have different lengths with an overwhelming probability, and their length are denoted as L_1 and L_2 respectively. More precisely, we search for a pair of integers i and j such that $d_1 + i \cdot L_1 = d_2 + j \cdot L_2$. Thus, the probability of a random pair (x_r, y_r) being the expected (x, y) is amplified by $\#C$ times, where $\#C$ is the maximum number of cycle loops that can be added. It contributes to improving preimage attacks on the XOR combiner. The complexity of our attack is $2^{5n/8}$, which is $2^{n/24}$ times lower than previous best-known complexity of $2^{2n/3}$ in [7]. We point out that the preimage message of our attack has a length of at least $2^{n/2}$ blocks, since the cycle length of an n-bit functional graph is $\Theta(2^{n/2})$.

Finally, we propose functional-graph-based second preimage attacks on Zipper hash. Differently from the XOR combiner and the concatenation combiner, the two passes of Zipper hash are sequential. Moreover, the second pass processes message blocks in a reversed order. These unique specifications bring extra degrees of freedom for the attacker. In details, after being linked to an internal state of the original message in the second pass, the first few blocks of our second preimage message are fixed. Note these blocks do not include the padding block of message length. As a result, we are always able to choose a length for second preimage message that optimizes the complexity. Moreover, when looking for a pair of nodes $(\check{x}, \check{y})$ reaching two predefined nodes of deep iterates $(\bar{x}, \bar{y})$ at a common distance, $\check{x}$ and $\check{y}$ are generated with different message blocks, since the message blocks are hashed in different orders in two passes. It enables us to launch a meet-in-the-middle procedure by using Joux's multi-collision when finding a pair of nodes $(\check{x}, \check{y})$, then the complexity of the attack is further reduced. If message length longer than $2^{n/2}$ is allowed, the complexity of our second preimage attack on Zipper hash is $2^{3n/5}$ for $L \geq 2^{2n/5}$, and $2^n/L$ for $0 < L < 2^{2n/5}$, where L is the block length of original message. Otherwise, the complexity of our attack is $2^{5n/8}$ for $2^{3n/8} < L \leq 2^{n/2}$, and $2^n/L$ for $0 \leq L < 2^{3n/8}$. We note these attacks are the first generic second-preimage attacks on Zipper hash to our best knowledge,[7] which solve an open problem proposed in [24].

Roadmap. Section 2 describes preliminaries. In Section 3, we provide a proof for Dinur's observation and further investigate properties of functional graph. Sections 4 and 5 present (second) preimage attacks on the XOR combiner and Zipper hash, respectively. Finally, we conclude the paper in Section 6.

---

[7] Assuming compression functions are weak, second-peimage attacks have been published on Zipper hash [5,19].

2 Preliminaries

2.1 Functional Graph

The functional graph (FG) of a random function f is defined by the successive iteration of this function. Explicitly, let f be an element of $\mathcal{F}_N$ which is the set of all mappings with a set N as both domain and range. The functional graph of f is a directed graph whose nodes are the elements $[0, \ldots, N-1]$ and whose edges are the ordered pairs $\langle x, f(x) \rangle$, for all $x \in [0, \ldots, N-1]$. If starting from any x_0 and iterating f, that is $x_1 = f(x_0), x_2 = f(x_1), \ldots$, we are going to find, before N iterations, a value x_j equal to one of $x_0, x_1, \ldots, x_{j-1}$. In this case, we say x_j is a *collision* and the path $x_0 \to x_1 \to \cdots \to x_{j-1} \to x_j$ connects to a *cycle* which describes the iteration structure of f starting from x_0. If we consider all possible starting points x_0, paths exhibit confluence and form into trees; trees grafted on cycles form components; a collection of components forms a functional graph [13].

Structure of FG has been studied for a long time, some parameters such as the number of components (*i.e.*, the number of connected components), the number of cyclic nodes (a node is cyclic if it belongs to a cycle), the number of terminal points (*i.e.*, nodes without preimage: $f^{-1}(x) = \emptyset$), the number of preimage points (*i.e.*, nodes with preimage), the expectation of tail length, the expectation of cycle length and rho-length have got accurate asymptotic evaluation [13], which are summarized below. A k-th iterate image point of f is an image point of the k-th iterate f^k of f.

Theorem 1 ([13]). *The expectations of parameters, number of components, number of cyclic points, number of terminal points, number of image points, and number of k-th iterate image points in a random mapping of size N have the asymptotic forms, as $N \to \infty$,*

1. *# Components* $\frac{1}{2} \log N = 0.5 \cdot n$
2. *# Cyclic nodes* $\sqrt{\pi N/2} \approx 1.2 \cdot 2^{n/2}$
3. *# Terminal nodes* $e^{-1} N \approx 0.37 \cdot 2^n$
4. *# Image points* $(1 - e^{-1})N \approx 0.62 \cdot 2^n$
5. *# k-th iterate image points* $(1 - \tau_k)N$

where the τ_k satisfies the recurrence $\tau_0 = 0$, $\tau_{k+1} = e^{-1+\tau_k}$.

Theorem 2 ([13]). *Seen from a random point (any of the N nodes in the associated functional graph is taken equally likely) in a random mapping of $\mathcal{F}_N$, the expectations of parameters tail length, cycle length, rho-length, tree size, component size, and predecessors size have the following asymptotic forms:*

1. *Tail length* (λ) $\sqrt{\pi N/8} \approx 0.62 \cdot 2^{n/2}$
2. *Cycle length* (μ) $\sqrt{\pi N/8} \approx 0.62 \cdot 2^{n/2}$
3. *Rho length* ($\rho = \lambda + \mu$) $\sqrt{\pi N/2} \approx 1.2 \cdot 2^{n/2}$
4. *Tree size* $N/3 \approx 0.34 \cdot 2^n$

5. *Component size* $2N/3 \approx 0.67 \cdot 2^n$
6. *Predecessors size* $\sqrt{\pi N/8} \approx 0.62 \cdot 2^{n/2}$

Theorem 3 ([13]). *Assuming the smoothness condition, the expected value of the size of the largest tree and the size of the largest connected component in a random mapping of* $\mathcal{F}_N$*, are asymptotically:*

1. *Largest tree:* $0.48 \cdot 2^n$*.*
2. *Largest component:* $0.75782 \cdot 2^n$*.*

Results from these theorems indicate that, in a random mapping, most of the points tend to be grouped together in a single giant component. This component might therefore be expected to have very tall trees and a large cycle.

2.2 XOR Combiner

The XOR combiner xors the outputs of two independent hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$, *i.e.* $\mathcal{H}_1(M) \oplus \mathcal{H}_2(M)$, which is depicted in Fig. 2 and Fig. 3.

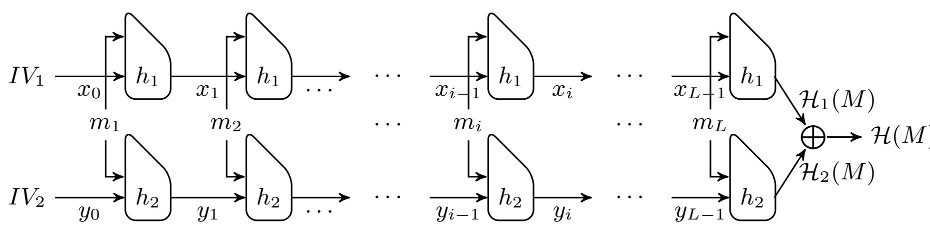

Fig. 2: The XOR combiner

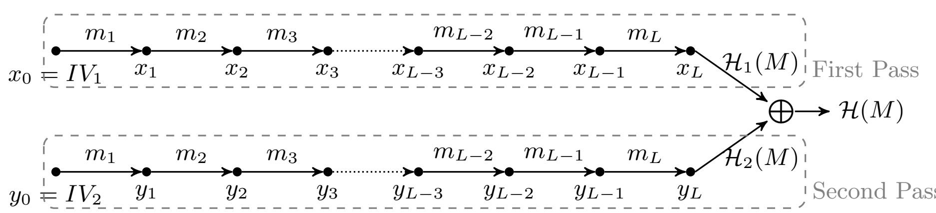

Fig. 3: Condensed graphical representation of the XOR combiner

2.3 Zipper Hash

Zipper hash is composed of two passes, denoted by $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively, operating on a single message. The two passes in Zipper hash are sequential, and the second pass operates the sequence of message blocks in a reversed order. The construction is depicted in Fig. 4 and Fig. 5.

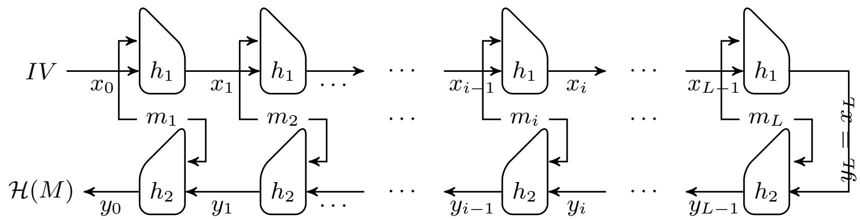

Fig. 4: The Zipper hash

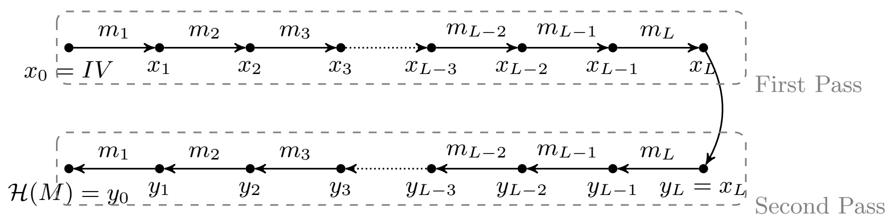

Fig. 5: Condensed graphical representation of the Zipper hash [19]

2.4 Joux's Multi-Collision [20]

In 2004, Joux [20] introduced multi-collisions on narrow-pipe Merkle-Damgård hash functions. Given a hash function $\mathcal{H}$, a multi-collision refers to a set of messages $\mathcal{M} = \{M_1, M_2, \ldots\}$ whose hash digests are all the same, *i.e.*, $\mathcal{H}(M_i) = \mathcal{H}(M_j)$ for any pair $M_i, M_j \in \mathcal{M}$. While the complexity is well-known to be birthday bound to find such a set when the size $|\mathcal{M}| = 2$, the computational complexity gradually approximates 2^n when the target size $|\mathcal{M}|$ increases. Utilizing the iterative nature of Merkle-Damgård structure, Joux's algorithm is able to find multi-collision of size 2^k with a complexity of $k \cdot 2^{n/2}$, *i.e.*, a complexity not much greater than that of finding a single collision. It works as follows. Given an iterative hash function $\mathcal{H}$ with compression function h, and an initial value x_0, one finds a pair of message (m_1, m'_1) such that $h(x_0, m_1) = h(x_0, m'_1) = x_1$ with a complexity of $2^{n/2}$. The process can be repeated to find (m_i, m'_i) such that $h(x_{i-1}, m_i) = h(x_{i-1}, m'_i) = x_i$ for $i = 2, 3, \ldots, k$ iteratively, as shown in Fig. 6. It is trivial to see the message set $\mathcal{M} = \{\overline{m}_1 \| \overline{m}_2 \| \cdots \| \overline{m}_k \mid \overline{m}_i = m_i \text{ or } m'_i \text{ for } i = 1, 2, \ldots, k\}$ form a multi-collision of size 2^k, and the overall complexity is $\mathcal{O}(k \cdot 2^{n/2})$.

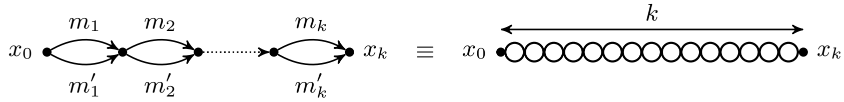

Fig. 6: Joux's multicollision structure and its condensed representation in R.H.S. [19]

2.5 Expandable Message [21]

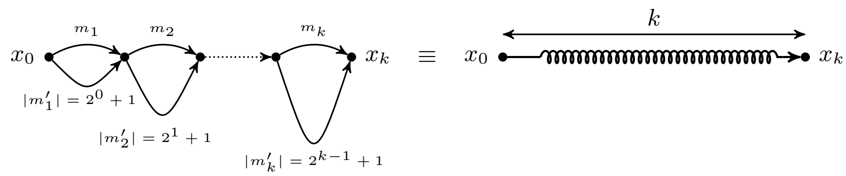

Fig. 7: The expandable message and its condensed representation in R.H.S. [19]

In 2005, Kelsey and Schneier [21] introduced the technique named *expandable message* to find second-preimages of Merkle-Damgård structure with a complexity of 2^{n-l}, instead of the long believed 2^n, for a given challenging message of about 2^l blocks. As depicted in Fig. 7, given a chaining value x_0 one finds a pair of message (m_1, m'_1) in time $2^{n/2}$ such that $h(x_0, m_1) = h(x_0, m'_1) = x_1$, and m_1 and m'_1 are of 1, and 2 blocks, respectively. The process can be repeated to find (m_i, m'_i) such that $h(x_{i-1}, m_i) = h(x_{i-1}, m'_i) = x_i$ for $i = 2, 3, \ldots, k$ iteratively, where m_i and m'_i are of 1, and $1 + 2^{i-1}$ blocks respectively. As a result, for each $t \in [k, k + 2^k - 1]$, there is a message of t blocks from the set $\mathcal{M} = \{\overline{m}_1 \| \overline{m}_2 \| \cdots \| \overline{m}_k \;\mid\; \overline{m}_i = m_i \text{ or } m'_i \text{ for } i = 1, 2, \ldots, k\}$. Note $h(x_0, M) = x_k$ for any $M \in \mathcal{M}$ and the overall complexity is $\mathcal{O}(k \cdot 2^{n/2} + 2^k)$. It is a special multi-collision, from which one can choose message of any desired block length in the range $[k, k + 2^k - 1]$.

Extension to two hash functions [7]. Dinur extends Kelsey and Schneier's technique [21] to build simultaneous expandable message on two hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$. Prior to that, Jha and Nandi propose a similar construction of an expandable message over two hash functions in the independent paper [19]. The main idea is, when building an expandable message on $\mathcal{H}_1$, to find two sets of $\mathcal{M} = \{m_i\}$ and $\mathcal{M}' = \{m'_i\}$ by Joux's multi-collision such that $h_1(x_{i-1}, m_i) = h_1(x_{i-1}, m'_i) = x_i$ for any $m_i \in \mathcal{M}$ and any $m'_i \in \mathcal{M}'$. Later, find a pair of $m_i \in \mathcal{M}$ and $m'_i \in \mathcal{M}'$ colliding on $\mathcal{H}_2$ that is $h_2(y_{i-1}, m_i) = h_2(y_{i-1}, m'_i) = y_i$. Hence, we find a pair of m_i and m'_i with carefully pre-determined lengths, colliding on both $\mathcal{H}_1$ and $\mathcal{H}_2$, with a complexity not much greater than that of finding a collision on a single hash function. The complexity is upper bounded by $L + n^2 \cdot 2^{n/2}$, where L is the maximum length that expandable message can produce. For completed description of the procedure, we refer to [7] or to Section 5.3 (slightly adapted due to specification of Zipper hash).

2.6 Dinur's Attack [7]

In [7], Dinur proposed second preimage attacks on the concatenation combiner and preimage attack on the XOR combiner, which are built on two independent

Merkle-Damgård hash functions. In this section, we briefly describe his attack on the XOR combiner $\mathcal{H}_1(M) \oplus \mathcal{H}_2(M)$.

The attack is based on functional graph. Fix a message m, and define $f_1(x) = h_1(x, m)$ and $f_2(y) = h_2(y, m)$, where h_1 and h_2 are the compression functions of $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively. In particular, it uses some special nodes, which are located deep in functional graphs defined by f_1 and f_2 and hence referred to as *deep iterates*. In other words, a deep iterate is a node that is reached after iterating f_1 (resp. f_2) many times.

Given a target hash digest V, the attack is composed of three main steps:

1. Build a simultaneous expandable message $\mathcal{M}$ for $\mathcal{H}_1$ and $\mathcal{H}_2$. It starts from the initial values (IV_1, IV_2) and ends at state $(\hat{x}, \hat{y})$.
2. Generate a set of tuples $\{(\bar{x}_1, \bar{y}_1, \bar{m}_1), (\bar{x}_2, \bar{y}_2, \bar{m}_2), \ldots\}$ such that $h_1(\bar{x}_i, \bar{m}_i) \oplus h_2(\bar{y}_i, \bar{m}_i) = V$, where $\bar{x}_i$ and $\bar{y}_i$ are chosen with a special property (being deep iterates).
3. Find a message $M_{\text{link}} = m_r \| m^d$, which links $(h_1(\hat{x}, m_r), h_2(\hat{y}, m_r))$ to some $(\bar{x}_i, \bar{y}_i)$ after iterating f_1 and f_2 by d times.

At the end, derive a message M_{L-2-d} with a block length of $L - 2 - d$ from the expandable message $\mathcal{M}$, and produce a preimage of V with a length L: $M = M_{L-2-d} \| M_{\text{link}} \| \bar{m}_i$.

$$(IV_1, IV_2) \xrightarrow{M_{L-2-d}} (\hat{x}, \hat{y}) \xrightarrow{M_{\text{link}}} (\bar{x}_i, \bar{y}_i) \xrightarrow{\bar{m}_i} (\mathcal{H}_1(M), \mathcal{H}_2(M)),$$

where $\mathcal{H}_1(M) \oplus \mathcal{H}_2(M) = V$ holds. The overall time complexity of Dinur's attack is optimally $2^{2n/3}$ obtained for $L = 2^{n/2}$.

The complexity advantage is gained thanks to two properties of deep iterates, which are listed below informally:

(i) it is easy to get a large set of deep iterates;
(ii) a deep iterate has a (relatively) high probability to be reached from an arbitrary starting node.

Property (i) contributes to the efficiency of Step 2, since one can find large sets of deep iterates in f_1 and f_2 independently, and then carry out a meet-in-the-middle procedure to find a set of tuples $\{(\bar{x}_i, \bar{y}_i, \bar{m}_i)\}$. Property (ii) contributes to the efficiency of Step 3. Thus, in order to estimate the complexity of the attack, it is necessary and important to study these two properties *quantitatively*.

- For property (i), $\Theta(2^t)$ iterates of depth 2^{n-t} can be collected with a complexity of 2^t by using Algorithm 1 for $t \geq n/2$;
- For property (ii), the author observes and experimentally verifies that the probability of a random pair (x_r, y_r) encountering d-th iterates $\bar{x}$ and $\bar{y}$ at a common distance (no larger than d) is $d^3/2^{2n}$ [7, Sec. 3.3].

However, the proof of property (ii) is remained incomplete, which is mainly because the sketch proof of Lemma 1 in [7] is not quite justified, when number of trails becomes too large, the dependency between different trails cannot be ignored anymore. In next section, we complete the proof for the observation of property (ii), which makes it solid for future usages in cryptography and other research fields.

Algorithm 1 Collect $\Theta(2^t)$ iterates of depth 2^{n-t} with a complexity of 2^t, for $t \geq n/2$

```
1:  procedure GEN(t)
2:      G ← ∅
3:      while |G| < 2^t do
4:          Chain ← ∅
5:          x ←$ {0, 1, ..., 2^n − 1} \ G
6:          while true do
7:              if x ∈ G or x ∈ Chain then
8:                  G ←merge Chain
9:                  go to line 3
10:             else
11:                 Chain ←insert x
12:                 x ← f(x)
13:             end if
14:         end while
15:     end while
16:     output G
17: end procedure
```

3 Functional Graph Revisited

3.1 Completing Proof of Dinur's Observation

In this section, we provide a formal proof on the probability of a pair of random starting points simultaneously reaching two deep iterates at a common distance, which is essential for the complexity advantage in [7].

Estimating the Number of k-th Iterates. To start with, we make an asymptotic estimate for the number of k-th iterates in the functional graph of a random mapping. As stated in Theorem 1, the expectation of the number of k-th iterate image points in a random mapping of size N has the asymptotic forms $(1-\tau_k)N$, as $N \to \infty$, where the τ_k satisfies the recurrence

$$\tau_0 = 0, \qquad \tau_{k+1} = e^{-1+\tau_k}.$$

Unfortunately this is an implicit expression. Here, we provide bounds on the explicit value of the number of k-th iterate image points, which will be used to complete the proof in the sequel.

Lemma 1. *The expectation of the number of k-th iterate image points $(1-\tau_k)N$ is bounded by*

$$\frac{1}{k}N < (1-\tau_k)N < \frac{2}{k}N,$$

where the τ_k satisfies the recurrence $\tau_0 = 0$, $\tau_{k+1} = e^{-1+\tau_k}$, and suppose $N = 2^n$ and $3 \le k \le \sqrt{N} = 2^{n/2}$.

Proof. Here, we use mathematical induction to prove this lemma:

Let $f_k = 1 - \tau_k$. Then, we need to prove that

$$1 < kf_k < 2.$$

Let $g_k = kf_k$, then it has

$$1 < g_3 = 3f_3 = 3(1 - e^{-1+e^{-1+e^{-1}}}) < 2 \tag{1}$$

For $k \ge 3$, suppose $1 < g_k < 2$, then we can prove $1 < g_{k+1} < 2$ as follows:

$$\begin{aligned}
(k+1)f_{k+1} = \quad & g_{k+1} = \quad (k+1)f_{k+1} \\
(k+1)(1-e^{-\frac{1}{k}g_k}) = \quad & g_{k+1} = \quad (k+1)(1-e^{-\frac{1}{k}g_k}) \\
(k+1)(1-e^{-\frac{1}{k}}) < \quad & g_{k+1} < \quad (k+1)(1-e^{-\frac{2}{k}}) \\
(k+1)(\frac{1}{k} - \frac{1}{2!}(\frac{1}{k})^2 + \frac{1}{3!}(\frac{1}{k})^3 \cdots) < \quad & g_{k+1} < \quad (k+1)(\frac{2}{k} - \frac{1}{2!}(\frac{2}{k})^2 + \frac{1}{3!}(\frac{2}{k})^3 \cdots) \\
(k+1)(\frac{1}{k} - \frac{1}{2k^2}) < \quad & g_{k+1} < \quad (k+1)(\frac{2}{k} - \frac{2}{k^2} + \frac{4}{3k^3}) \\
1 + \frac{1}{2k} - \frac{1}{2k^2} < \quad & g_{k+1} < \quad 2 + \frac{2}{k}(1 - \frac{k+1}{k} + \frac{2(k+1)}{3k^2}) \\
1 + \frac{1}{k}(\frac{1}{2} - \frac{1}{2k}) < \quad & g_{k+1} < \quad 2 + \frac{2}{k}(-\frac{1}{k} + \frac{2(k+1)}{3k^2}) \\
1 + \frac{1}{k}\cdot\frac{k-1}{2k} < \quad & g_{k+1} < \quad 2 + \frac{2}{k}\cdot\frac{-k+2}{3k^2} \\
1 < \quad & g_{k+1} < \quad 2
\end{aligned}$$

The last line holds as $k \ge 3$. ☐

We refer to Rényi and Szekeres's paper [34] in which one can find a consistent yet more precise estimate on the ratio τ_k. It suggests that $\lim_{k\to\infty} kf_k = 2$. Thus, one can get the following analogous lemma.

Lemma 2. *The expectation of number of k-th iterate image points is*

$$(1-\tau_k) \times N \approx (\frac{2}{k} - \frac{2}{3}\frac{\log k}{k^2} - \frac{c}{k^2} - \cdots) \times N \approx 2^{n-\log_2(k)+1},$$

where the τ_k satisfy the recurrence $\tau_0 = 0$, $\tau_{k+1} = e^{-1+\tau_k}$, c is a certain constant, and suppose $N = 2^n$ and $k \le \sqrt{N} = 2^{n/2}$.

Proof. Refer to Equation (3.9) and (3.10) in [34].

Remark 1. We note that Lemmas 1 and 2 imply a theorem about the entropy loss shown in [8] when each f_i's are all identical and one takes $2^s = k$ as iterate depth, see Theorem 4. Moreover, these Lemmas show that the entropy loss 2^{n-s} is not only an upper bound, *i.e.*, $\tilde{O}(2^{n-s})$, but also a tight bound *i.e.*, $\Theta(2^{n-s})$.

Theorem 4 (Entropy loss [8]). *Let $s \leq n/2$ be a non-negative integer. Let $f_1, f_2, \ldots, f_{2^s}$ be a fixed sequence of random functions over the set of 2^n elements, and $g_{2^s} \triangleq f_{2^s} \circ f_{2^{s-1}} \circ \cdots \circ f_2 \circ f_1$ (with the f_i being either all identical, or independently distributed). Then, the images of g_{2^s} contains at most $\tilde{O}(2^{n-s})$ points.*

On Probability of Simultaneously Reaching Two k-th Iterates. Next, we consider the probability of simultaneously reaching two k-th iterate image points from two random starting points in the functional graphs of two independent random mappings.

Theorem 5. *Let f be a random mapping, k and d ($d \leq k \leq 2^{n/2}$) be two positive integers. Then*

$$p_{dk} = \Pr\left(f^d(x_r) = x_k\right) > \frac{d}{2} \cdot 2^{-n}, \tag{2}$$

where x_k is a predefined k-th iterate image point that is selected by choosing a random value x_0 and applying f iteratively by k times: $x_k = f^k(x_0)$, and x_r is a random starting point.

Proof. From Lemma 1, the number of d-th iterate image point of h is less than $\frac{2}{d} \cdot 2^n$. Because that x_k is a point with depth k and $k \geq d$, x_k is also a d-th iterate image point. Therefore, the probability that f^d operating on a random starting point x_r hits the predefined image point x_k is greater than $d/2 \cdot 2^{-n}$. Clearly it holds if the image of f^d is uniformly distributed. We point out that if the image distribution is not uniformly distributed, the probability will be even higher. Note that x_k is selected as $x_k = f^k(x_0)$ with a random x_0 and $k \geq d$. Hence, it is more likely an image that can be reached after d iterates with a higher probability. □

Remark 2. We remark that Theorem 5 is a restatement and a slight improvement of Lemma 1 in [7]. The sketch proof of Lemma 1 in [7] raises the issue discussed by Dinur in Appendix B of [7]. That is, when we have evaluated more than $2^n/D$ nodes and consider the probability of a new chain colliding with the original chain which is of D length, we cannot ignore the birthday paradox and thus the outcome of new trials cannot be analyzed probabilistically like that analyzed in the proof of Lemma 1 in [7]. The proof of Theorem 5 utilizes the property of deep iterates in functional graph of a random mapping. This property of deep iterates is a statistical property of functional graph of random mappings, which originally results from analytic combination theories [13]. It overcomes the difficulty to analyze the probability when more starting points are selected by assuming that the distribution of images of f^d is uniformly distributed, or if not uniformly distributed then mapping to a deeper-iterate with a higher probability. That enables us to provide an estimate on the probability of the outcome of new trials and thus complete the theoretical analysis of the complexity of the attack.

The following theorem is consistent with Dinur's observation on the probability of starting from two random points and encountering two deep iterates at a common distance which is deduced from Lemma 1 in [7]. In the following, we restate it and prove it using Theorem 5 as follows.

Theorem 6. *Let f_1 and f_2 be two random mappings, k be a positive integer and $k \leq 2^{n/2}$. Then*

$$\Pr\left(f_1^d(x_r) = x_k \text{ and } f_2^d(y_r) = y_k \mid d \in \mathbb{Z} \text{ and } d \leq k\right) > \frac{1}{12}k^3 \cdot 2^{-2n} + o(k^3 \cdot 2^{-2n}), \tag{3}$$

where x_k (resp. y_k) is a predefined k-th iterate image point and x_r (resp. y_r) is a random starting point in the functional graph of f_1 (resp. f_2).

Proof. Considering f_1 and f_2 are two independent random mappings, it has that $\Pr\left(f_1^d(x_r) = x_k \text{ and } f_2^d(y_r) = y_k\right) = \Pr\left(f_1^d(x_r) = x_k\right) \cdot \Pr\left(f_2^{\circ d}(y_r) = y_k\right)$. Thus, from Theorem 5

$$\begin{aligned}
&\Pr\left(f_1^d(x_r) = x_k \text{ and } f_2^d(y_r) = y_k \mid d \in \mathbb{Z} \text{ and } d \leq k\right) &(4)\\
&= \sum_{d=1}^{k} \Pr\left(f_1^d(x_r) = x_k \text{ and } f_2^d(y_r) = y_k\right) &(5)\\
&= \sum_{d=1}^{k} \Pr\left(f_1^d(x_r) = x_k\right) \cdot \Pr\left(f_2^d(y_r) = y_k\right) &(6)\\
&> \sum_{d=1}^{k} \left(\frac{d}{2} \cdot 2^{-n}\right)^2 &(7)\\
&= \frac{1}{12}k^3 \cdot 2^{-2n} + o(k^3 \cdot 2^{-2n}). &(8)
\end{aligned}$$

□

Up to this point, we have provided a formal proof of Dinur's observation. One might get more precise estimation on this probability by applying Lemma 2.

3.2 Cyclic Node and Multi-Cycles

In this section, we study a property of cyclic nodes within functional graph of a random mapping. Each cyclic node in a functional graph defined by f loops along the cycle when computed by f iteratively, and goes back to itself after a (multi-) cycle-length number of function calls. This property can be utilized to provide extra degrees of freedom, when estimating the distance of other nodes to a cyclic node in the functional graph, *i.e.*, it can be expanded to a set of discrete values by using multi-cycles. For example, let x and x' be two nodes in a component of the functional graph defined by f, x is a cyclic node, and the cycle length of the component is denoted as L. Clearly there exists a path from x' to x as they are in the same component, and the path length is denoted as d. Then we have

$$f^d(x') = x; \quad f^L(x) = x \quad \Longrightarrow \quad f^{(d+i\cdot L)}(x') = x \quad \text{for any positive integer } i.$$

Suppose it is limited to use at most t cycles. Then the distance from x' to x is expanded to a set of $t+1$ values $\{d + i \cdot L \mid i = 0, 1, 2, ..., t\}$.

Now let us consider a special case of reaching two deep iterates from two random starting nodes: *select two cyclic nodes within the largest components in the functional graphs as the deep iterates*. More specifically, let two functional graphs be defined by f_1 and f_2. Let $\bar{x}$ and x_r be two nodes in a common largest component of functional graph defined by f_1, where $\bar{x}$ is a cyclic node. Let L_1 denote the cycle length of the component and d_1 denote the path length from x_r to $\bar{x}$. Similarly, we define notations $\bar{y}$, y_r, L_2 and d_2 in functional graph of f_2. We are interested in the probability of linking x_r to $\bar{x}$ and y_r to $\bar{y}$ at a common distance. Thanks to the usage of multiple cycles, the distance values from x_r to $\bar{x}$ and from y_r to $\bar{y}$ can be selected from two sets $\{d_1 + i \cdot L_1 \mid i = 0, 1, 2, \ldots, t\}$ and $\{d_2 + j \cdot L_2 \mid j = 0, 1, 2, \ldots\ t\}$, respectively. Hence, as long as there exists a pair of integers (i, j) such that $0 \le i, j \le t$ and $d_1 + i \cdot L_1 = d_2 + j \cdot L_2$, we get a common distance $d = d_1 + i \cdot L_1 = d_2 + j \cdot L_2$ such that

$$f_1^d(x_r) = \bar{x}, \qquad\qquad f_2^d(y_r) = \bar{y}.$$

Next, we evaluate the probability amplification of reaching $(\bar{x}, \bar{y})$ from a random pair (x_r, y_r) at the same distance. Without loss of generality, we assume $L_1 \le L_2$. Let ΔL be $\Delta L = L_2 \mod L_1$. Then, it has that

$$\begin{aligned} d_1 + i \cdot L_1 &= d_2 + j \cdot L_2 && \Longrightarrow \\ d_1 - d_2 &= j \cdot L_2 - i \cdot L_1 && \Longrightarrow \\ (d_1 - d_2) \mod L_1 &= j \cdot \Delta L \mod L_1 \end{aligned}$$

Letting j range over all integer values in internal $[0, t]$, we will collect a set of $t+1$ values $\mathcal{S} = \{j \cdot \Delta L \mod L_1 \mid j = 0, 1, \ldots, t\}$[8]. Since $d_1 = \mathcal{O}(2^{n/2})$, $d_2 = \mathcal{O}(2^{n/2})$ and $L_1 = \Theta(2^{n/2})$, it has $|d_1 - d_2| = \mathcal{O}(L_1)$, and we assume $|d_1 - d_2| < L_1$ by ignoring the constant factor. Therefore, for a randomly sampled pair (x_r, y_r) that encounter $(\bar{x}, \bar{y})$, we are able to derive a pair of (i, j) such that $d_1 + i \cdot L_1 = d_2 + j \cdot L_2$, as long as their distance bias $d_1 - d_2$ is in the set $\mathcal{S}$.

[8] With very low probability L_1 and L_2 are not co-prime, and large t will result in repeated values.

In other words, we are able to *correct such a distance bias by using multi-cycles.* Thus, the probability of reaching $(\bar{x}, \bar{y})$ from a random pair (x_r, y_r) at a common distance is amplified by roughly t times, which is the maximum number of cycles used.

This property of cyclic nodes in functional graph can be utilized to improve preimage attacks on the XOR combiner, which is presented in next sections. The set $\mathcal{S}$ is referred to as the set of *correctable distance bias* hereafter.

4 Improved Preimage Attack on XOR Combiner

4.1 Attack Overview

Firstly, we recall previous preimage attack on the XOR combiner [7] introduced in Section 2.6. We name $(\bar{x}_i, \bar{y}_i)$'s as *target* node pairs. Clearly the larger the number of target node pairs (generated at Step 2) is, the higher the probability of a random node pair $(x_r = h_1(\hat{x}, m_r), y_r = h_2(\hat{y}, m_r))$ reaching a target node pair $(\bar{x}_i, \bar{y}_i)$ (at Step 3) at a common distance becomes. Hence, a complexity tradeoff exists between Steps 2 and 3. The optimal complexity is obtained by balancing Step 2 and Step 3.

In this section, we use cyclic nodes and multi-cycles to improve preimage attack on the XOR combiner. More specifically, if a target node pair $(\bar{x}, \bar{y})$ are both cyclic nodes within the largest components in two functional graphs respectively, the probability of a random pair $(x_r = h_1(\hat{x}, m_r), y_r = h_2(\hat{y}, m_r))$ reaching $(\bar{x}, \bar{y})$ at a common distance is amplified by $\#C$ times, the maximum number of cycles that can be used, by using the set of correctable distance bias as stated in Section 3.2. Moreover, such a probability amplification comes with almost no increase of complexity at Step 2, which leads to a new complexity tradeoff between Steps 2 and 3. Thus, the usage of cyclic nodes and multi-cycles enables us to reduce the computational complexity of preimage attacks on the XOR combiner.

Here we briefly list the *main* steps of our preimage attack on the XOR combiner.

Step A. Build a simultaneous expandable message $\mathcal{M}$ for $\mathcal{H}_1$ and $\mathcal{H}_2$ ending with $(\hat{x}, \hat{y})$.

Step B. Collect cyclic nodes within the largest components in functional graphs of $f_1 = h_1(\cdot, m)$ and $f_2 = h_2(\cdot, m)$ with a fixed m, and compute the set of correctable distance bias

$$\mathcal{S} = \{i \cdot \Delta L \mod L_1 \mid i = 0, 1, \ldots, \#C\},$$

where L_1 and L_2 are cycle length of the largest components in the functional graphs of f_1 and f_2 respectively and $\Delta L = L_2 - L_1 \mod L_1$.

Step C. Find a set of tuples $\{(\bar{x}_1, \bar{y}_1, \bar{m}_1), (\bar{x}_2, \bar{y}_2, \bar{m}_2), \ldots\}$ such that $\bar{x}_i$'s and $\bar{y}_j$'s are cyclic nodes within the largest components in functional graphs of f_1 and f_2 respectively, and $h_1(\bar{x}_i, \bar{m}_i) \oplus h_2(\bar{y}_i, \bar{m}_i) = V$, where V is the target hash digest.

Step D. Find a message $M_{\text{link}} = m_r \| m^d$ that links $(\hat{x}, \hat{y})$ to some $(\bar{x}_i, \bar{y}_i)$. For each pair $(x_r = h_1(\hat{x}, m_r), y_r = h_2(\hat{y}, m_r))$ that encounters $(\bar{x}_i, \bar{y}_i)$, compute the distance difference and examine whether it belongs to $\mathcal{S}$.

Up to now, we are able to derive a message M_e from expandable message with an appropriate length, and produce a preimage message M:

$$(IV_1, IV_2) \xrightarrow{M_e} (\hat{x}, \hat{y}) \xrightarrow{M_{\text{link}}} (\bar{x}_i, \bar{y}_i) \xrightarrow{\bar{m}_i} (\mathcal{H}_1(M), \mathcal{H}_2(M)) : \mathcal{H}_1(M) \oplus \mathcal{H}_2(M) = V$$

By balancing the complexities of these steps, we obtain an optimal complexity of $2^{5n/8}$.

A completed description of attack procedure and complexity evaluation is provided in next sections. We point out the length of our preimage is at least $2^{n/2}$ block long due to the usage of (multi-) cycles.

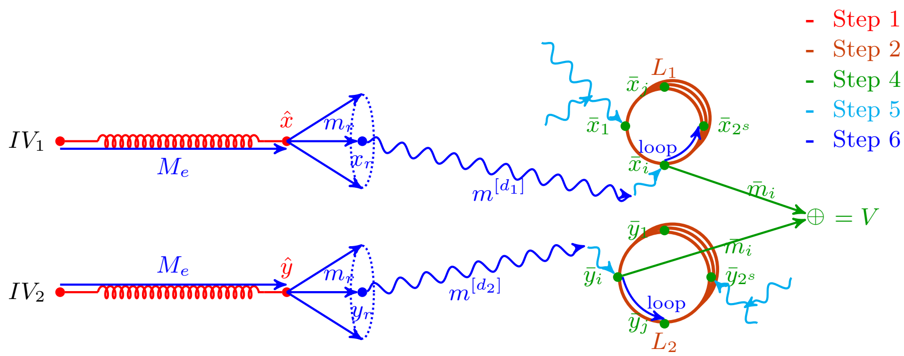

Fig. 8: Preimage attack on the XOR hash combiner $\mathcal{H}_1(M) \oplus \mathcal{H}_2(M)$

4.2 Attack Procedure

Denote by V the target hash digest. Suppose the attacker is going to produce a preimage message with a length L. The value of L will be discussed later. The attack procedure is described below.

1. Build a simultaneous expandable message structure $\mathcal{M}$ ending with a pair of state $(\hat{x}, \hat{y})$ such that for each positive integer i of an integer interval, there is a message M_i with a block length i in $\mathcal{M}$ that links (IV_1, IV_2) to $(\hat{x}, \hat{y})$:

$$h_1^i(IV_1, M_i) = \hat{x}, \qquad h_2^i(IV_2, M_i) = \hat{y}.$$

   Refer to Section 2.5 and [7] for more descriptions of the procedure.

2. Fix a single-block message m, and construct two n-bit to n-bit random mappings as $f_1(x) = h_1(x, m)$ and $f_2(y) = h_2(y, m)$. Repeat the cycle search several times, and find all the cyclic nodes within the largest components in the functional graphs defined by f_1 and f_2. Denote their cycle lengths as L_1 and L_2, and the two sets of cyclic nodes as $\{x_1, x_2, \ldots, x_{L_1}\}$ and $\{y_1, y_2, \ldots, y_{L_2}\}$, and store them in tables $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively.
3. Without loss of generality, assume $L_1 \leq L_2$. Compute $\#C = \lfloor L/L_1 \rfloor$ as the maximum number of cycles that can be used to correct distance bias. Compute $\Delta L = L_2 \mod L_1$, and then compute the set of correctable distance bias: $\mathcal{S} = \{i \cdot \Delta L \mod L_1 \mid i = 0, 1, 2, \ldots, \#C\}$.
4. Find a set of 2^s tuples $(\bar{x}, \bar{y}, \bar{m})$ such that $h_1(\bar{x}, \bar{m}) \oplus h_2(\bar{y}, \bar{m}) = V$. The search procedure is described as follows.
   (a) Initialize a table $\mathcal{T}_3$ as empty.
   (b) Select a random single-block message $\bar{m}$.
   (c) Compute $h_1(x_i, \bar{m})$ for all x_i's in $\mathcal{T}_1$, and store them in a table $\mathcal{T}_4$.
   (d) For each y_j in $\mathcal{T}_2$, compute $h_2(y_j, \bar{m}) \oplus V$, and match it to the elements in $\mathcal{T}_4$. If it is matched to some $h_1(x_i, \bar{m})$, that is
   $$h_2(y_j, \bar{m}) \oplus V = h_1(x_i, \bar{m}) \quad \Longrightarrow \quad h_1(x_i, \bar{m}) \oplus h_2(y_j, \bar{m}) = V,$$
   store $(x_i, y_j, \bar{m})$ in $\mathcal{T}_3$.
   (e) If $\mathcal{T}_3$ contains less than 2^s elements, goto step 4(b) and repeat the search procedure.

   Denote the stored tuples in $\mathcal{T}_3$ as $\{(\bar{x}_1, \bar{y}_1, \bar{m}_1), \ldots, (\bar{x}_{2^s}, \bar{y}_{2^s}, \bar{m}_{2^s})\}$. Moreover, $\bar{x}_i$'s and $\bar{y}_j$'s are called *target nodes* in functional graphs of f_1 and f_2 respectively.
5. Run Algorithm 1 with a parameter t to develop 2^t nodes in the functional graph of f_1 (resp. f_2), and store them in a table $\mathcal{T}_{4x}$ (resp. $\mathcal{T}_{4y}$). Moreover,
   (a) Store at each node its distance from a particular target node (say target node $\bar{x}_1$ (resp. $\bar{y}_1$), similar to phase 3 in Section 3.3 of [7]), together with its distance from the cycle (i.e. its height, similar to phase 3 in Section 5 of [33]).
   (b) Store the distance of other target nodes $\bar{x}_i$ (resp. $\bar{y}_i$) to this particular target node $\bar{x}_1$ (resp. $\bar{y}_1$) in a table $\mathcal{T}_{3x}$ (resp. $\mathcal{T}_{3y}$) by iterating f_1 (resp. f_2) along the cycle.
   (c) Thus, when the distance of a node from the particular target node and that from the cycle is known from $\mathcal{T}_{4x}$ (resp. $\mathcal{T}_{4y}$), the distances of this node from all the other target nodes can be immediately deduced from $\mathcal{T}_{3x}$ (resp. $\mathcal{T}_{3y}$). Specifically, suppose the distance of a node x_r from $\bar{x}_1$ is d_1 and its height is e_1, and suppose the distance of a target node $\bar{x}_i$ from $\bar{x}_1$ is d_i, then the distance of x_r from $\bar{x}_i$ is $d_1 - d_i$ if $d_i \leq (d_1 - e_1)$, and $L_1 - d_i + d_1$ if $d_i > (d_1 - e_1)$.
6. Find a message M_{link} that links $(\hat{x}, \hat{y})$ to a pair of target nodes $(\bar{x}_i, \bar{y}_i)$ in $\mathcal{T}_3$. We search for such a linking message among a set of special messages: $M_{\text{link}} = m_r \| m \| m \| \cdots \| m$, where m_r is a random single-block message, and m is the fixed message at Step 2. The search procedure is as follows.

(a) Select a random m_r, and compute $x_r = h_1(\hat{x}, m_r)$ and $y_r = h_2(\hat{y}, m_r)$;
(b) Compute a chain by iteratively applying f_1 (resp. f_2) to update x_r (resp. y_r), until either of the following two cases occurs.
   – The chain length reaches 2^{n-t}. In this case, goto step 6(a);
   – The chain encounters a node stored in $\mathcal{T}_{4x}$ (resp. $\mathcal{T}_{4y}$). Compute the distance of x_r (resp. y_r) to every target node $\bar{x}_i$ (resp. $\bar{y}_i$) as described in step 5(c), and denote it as dx_i (resp. dy_i).
(c) Examine whether $dx_i - dy_i \mod L_1$ is a correctable distance difference in $\mathcal{S}$. If it is, derive the corresponding j and k such that $dx_i + j \cdot L_1 = dy_i + k \cdot L_2$ holds. Let p be $p = dx_i + j \cdot L_1 = dy_j + k \cdot L_2$, and then $M_{\text{link}} = m_r \| m^p$. Otherwise, goto step 6(a).

7. Derive a message M_{L-2-p} with a block length of $L-2-p$ from the expandable message $\mathcal{M}$.
8. Produce a preimage M of the target hash digest V as

$$M = M_{L-2-p} \| M_{\text{link}} \| \bar{m}_i = M_{L-2-p} \| m_r \| m^p \| \bar{m}_i.$$

4.3 Attack Complexity

This subsection evaluates the attack complexity. In particular, we note that we ignore the constant and polynomial factors for the simplicity of description.

- Step 1: $L + n^2 \cdot 2^{n/2}$ (refer to Sect 2.5);
- Step 2: $2^{n/2}$;
- Step 3: $L/L_1 \approx 2^{-n/2} \cdot L$;
- Step 4: $2^{s+n/2}$;
  One execution of the search procedure takes a complexity of $L_1 + L_2$, and contributes to $L_1 \cdot L_2$ pairs. As $L_1 \cdot L_2 = \Theta(2^n)$, one tuple can be obtained by a constant number of executions. Hence, the number of necessary executions is $\Theta(2^s)$, and the complexity of this step is $\Theta(2^{s+n/2})$.
- Step 5: $2^t + 2^{n/2}$;
  The complexity of developing 2^t nodes and computing their distance to a particular target node is 2^t (refer to Algorithm 1 and step 5(a)). The complexity to compute the distance of all the other target nodes to the particular target node is upper bounded by $2^{n/2}$ (refer to the number of cyclic nodes in Theorem 1). Hence, the complexity of this step is $2^t + 2^{n/2}$.
- Step 6: $2^{2n-t-s}/L$;
  One execution of the search procedure needs a time complexity of 2^{n-t}. Clearly a constant factor of both of the two chains encounter nodes stored in $\mathcal{T}_{4x}$ and $\mathcal{T}_{4y}$ which are of size 2^t. We mainly need to evaluate the probability of deriving a common distance for each chain. For every pair of target nodes $(\bar{x}_i, \bar{y}_i)$, the value of $dx_i - dy_i$ is equal to a correctable distance bias in $\mathcal{S}$ with a probability of $\#C \cdot 2^{-n/2} \approx L \cdot 2^{-n}$. Since there are 2^s pairs of target nodes, the success probability of each chain is $L \cdot 2^{s-n}$. Hence, the total number of chains is $2^{n-s}/L$, and the complexity of this step is $2^{n-t} \cdot 2^{n-s}/L = 2^{2n-t-s}/L$.

- Steps 7 and 8: $\mathcal{O}(L)$.

The overall complexity is computed as

$$L + 2^{s+n/2} + 2^t + 2^t + 2^{n/2} + \frac{2^{2n-t-s}}{L},$$

where the complexities of steps 2, 7 and 8 are ignored.

Now we search for parameters L, t and s that give the lowest complexity. Firstly, we balance the complexities between Step 1 and Step 6, that gives

$$L = \frac{2^{2n-t-s}}{L} \quad \Longrightarrow \quad L = 2^{n-t/2-s/2}$$

Hence, the total complexity becomes (ignoring constant factors)

$$2^{n-t/2-s/2} + 2^{s+n/2} + 2^t + 2^{n/2}.$$

By balancing the complexities, we have that setting parameters $t = 2^{5n/8}$ and $s = 2^{n/8}$ contributes to the lowest complexities: $2^{5n/8}$. In the setting, we produce a preimage message with a length of $L = 2^{n-t/2-s/2} = 2^{5n/8}$.

5 Second Preimage Attacks on Zipper Hash

In this section, we give a second preimage attack on Zipper hash, which is also applicable for *idealized* compression functions.

5.1 Attack Overview

Given a message $M = m_1 \| m_2 \| \cdots \| m_L$, the second preimage attack on Zipper hash is to find another message M' such that $\mathcal{H}_2(\mathcal{H}_1(IV, M), \overleftarrow{M}) = \mathcal{H}_2(\mathcal{H}_1(IV, M'), \overleftarrow{M'})$, where $\overleftarrow{M}$ is a message generated by reversing the order of message blocks of M (we call $\overleftarrow{M}$ the reverse of M for simplicity), *i.e.*, $\overleftarrow{M} = m_L \| m_{L-1} \| \cdots \| m_2 \| m_1$, and $\overleftarrow{M'}$ is the reverse of M'.

In contrast to the attack against XOR combiner, here we make use of a single pair of target α-nodes $(\bar{x}, \bar{y})$, that are the roots of the largest trees within the largest components in functional graphs defined by h_1 and h_2 and a fixed single-block message value m, and then proceed as follows.

Step A. Compute a multi-collision $\mathcal{M}_1$ (resp. $\mathcal{M}_2$) from $\bar{x}$ (resp. $\bar{y}$) as the starting value to an ending value denoted as $\hat{x}$ (resp. $\hat{y}$).

Step B. Build a simultaneous expandable message $\mathcal{M}_e$ across the two passes, starting from $\hat{x}$ to an ending value denoted as $\tilde{y}$.

Step C. Find $\bar{m}$ linking $\hat{y}$ to one of the chaining values y_q of the second pass of the original message, and then compute from x_q with $\bar{m}$ to an internal value denoted as $\tilde{x}$ in the first pass.

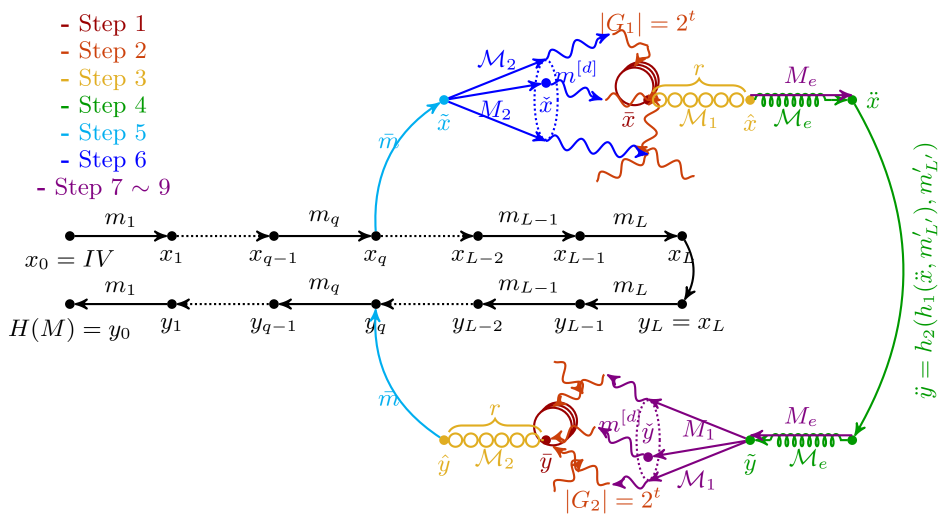

Fig. 9: Second preimage attack on Zipper hash

Step D. Exploit the messages of $\mathcal{M}_2$ and $\mathcal{M}_1$ to link $\tilde{x}$ and $\tilde{y}$ to $\bar{x}$ and $\bar{y}$, respectively, at a common distance.

Finally, we just need to derive a message with a suitable length from $\mathcal{M}_e$ to contribute to a second preimage message. There are two main differences between the attack on Zipper hash and the attack on the XOR combiner in Section 4. One is that linking $\tilde{x}$ to $\bar{x}$ and linking $\tilde{y}$ to $\bar{y}$ can be carried out independently, resulting in a meet-in-the-middle like effect. The other is that the message length is embedded inside the expandable message $\mathcal{M}_e$, which enables us to choose the length of second preimage message to optimize the complexity. Details of the attack are presented in the next sections.

5.2 Attack Procedure

In the attack procedure below, we omit the description of using multi-cycles to correct distance bias for the simplicity of description. We note that multi-cycles should be used at Steps 6 and 7, which provides extra degrees of freedom, in the case that the message length is allowed beyond the birthday bound.

1. Fix an arbitrary single-block message value m, and construct $f_1(\cdot) = h_1(\cdot, m)$ and $f_2(\cdot) = h_2(\cdot, m)$. Repeat the cycle search several times to locate the largest tree and corresponding α-node in functional graph of f_1 (resp. f_2) and denote it by $\bar{x}$ (resp. $\bar{y}$).
2. Run Algorithm 1 with a parameter t to develop 2^t nodes, compute and store their distance from $\bar{x}$ (resp. $\bar{y}$) in functional graph of f_1 (resp. f_2). Store these nodes of f_1 (resp. f_2) in the data structure G_1 (resp. G_2). The role of

G_1 (resp. G_2) is to reduce the number of evaluations of f_1 (resp. f_2) to find the distance of a random starting node from the target node $\bar{x}$ (resp. $\bar{y}$) at Step 6 and Step 7. This is similar to the lookahead procedure of phase 3 in Section 3.3 of [7].

3. Build a Joux's multi-collision of size 2^r for h_1 (resp. h_2) starting from $\bar{x}$ (resp. $\bar{y}$) and ending at a node $\hat{x}$ (resp. $\hat{y}$). Denote the multi-collision message set by $\mathcal{M}_1$ (resp. $\mathcal{M}_2$).
4. Construct a simultaneous expandable message $\mathcal{M}_e$ across the two hash functions, which starts from the node $\hat{x}$ in the first pass and ends at a node $\tilde{y}$ in the second pass. The details of constructing such an expandable message is provided in Section 5.3.
5. Find a single-block $\bar{m}$ that links $\hat{y}$ to some internal state y_q of the second pass on computing the original message M. The search procedure is trivial and hence omitted. Compute $\tilde{x} = h_1(x_q, \bar{m})$.
6. For each message M_2 in $\mathcal{M}_2$,
   (a) Compute $\check{x} = h_1^r(\tilde{x}, M_2)$;
   (b) Compute a chain $\boldsymbol{x}$ by applying f_1 to update $\check{x}$ iteratively until up to a maximum length 2^{n-t} or until it hits G_1. In the latter case, compute the distance d_1 of $\check{x}$ to $\bar{x}$, and store (d_1, M_2) in a table $\mathcal{T}_1$.
7. For each message M_1 in $\mathcal{M}_1$,
   (a) Compute $\check{y} = h_2^r(\tilde{y}, M_1)$;
   (b) Compute a chain $\boldsymbol{y}$ by applying f_2 to update $\check{y}$ up to a maximum length 2^{n-t} or until it hits G_2. In the latter case, compute the distance d_2 of $\check{y}$ to $\bar{y}$, and store (d_2, M_1) in a table $\mathcal{T}_2$.
8. Find (d_1, M_2) in $\mathcal{T}_1$ and (d_2, M_1) in $\mathcal{T}_2$ with $d_1 = d_2$. The search is a meet-in-the-middle procedure to match elements between $\mathcal{T}_1$ and $\mathcal{T}_2$.
9. Derive a message M_e with a block length $L' - q - 1 - r - d_2 - r$ from $\mathcal{M}_e$, where L' is the length of the constructed second preimage. Construct a message $M' = m_1\|m_2\|\cdots\|m_q\|\bar{m}\|M_2\|m^{[d_2]}\|M_1\|M_e$ and output M' as a second preimage.

5.3 Step 4: Constructing an Expandable Message

The constructing method is similar with that proposed in [7] with slight modifications. Detailed steps are as follows and the constructing process is depicted in Fig. 10, where C is set as a constant such that $C \approx n/2 + \log(n)$:

1. $x'_0 \leftarrow \hat{x}$
2. For $i \leftarrow 1,\ 2, \cdots, C - 1 + k$:
   (a) Build a 2^{C-1} standard Joux's multi-collision in h_1 starting from x'_{i-1}, denote its final endpoint by sp_i.
   (b) Compute $xp_i = h_1(sp_i, \mathbf{0})$, where $\mathbf{0}$ is an all zero message of size s blocks, where $s = i$ if $i \leq C - 1$ and $s = C2^{i-(C-1)-1}$ if $C - 1 < i \leq C - 1 + k$.
   (c) Find a collision $h_1(sp_i, m_i) = h_1(xp_i, m'_i)$ with single block messages $m_i,\ m'_i$. Denote the collision by x'_i.
   (d) We get a multi-collision in h_1 with 2^C messages that map x'_{i-1} to x'_i.

i. Out of these messages, 2^{C-1} are of length b (obtained by combine one of the 2^{C-1} Joux's multi-collisions with m_i) and we denote this set of messages by SS_i, where $b = C$.

ii. Out of these messages, 2^{C-1} are of length b (obtained by combine one of the 2^{C-1} Joux's multi-collisions with $\mathbf{0}\|m'_i$) and we denote this set of messages by SL_i, where $b = C + i$ if $i \leq C - 1$ and $b = C(2^{i-(C-1)-1} + 1)$ if $C - 1 < i \leq C - 1 + k$.

3. Denote the last collision state x'_{C-1+k} by $\ddot{x}$, and compute $\ddot{y} = h_2(h_1(\ddot{x}, m'_{L'}), m'_{L'})$, where $m'_{L'}$ is a message block padded with the length L' of the second preimage.
4. $y'_{C-1+k} \leftarrow \ddot{y}$, $MS \leftarrow \emptyset$, $ML \leftarrow \emptyset$.
5. For $i \leftarrow C-1+k,\ C-1+k-1,\ \ldots,\ 2,\ 1$:
   (a) For each $\boldsymbol{ms}_i \in SS_i$, compute $u_i = h_2(y'_i, \overleftarrow{\boldsymbol{ms}}_i)$ where $\boldsymbol{ms}_i = ms_{i,1}\|ms_{i,2}\|\ldots\|ms_{i,C-1}\|ms_{i,C}$ and $\overleftarrow{\boldsymbol{ms}}_i = ms_{i,C}\|ms_{i,C-1}\|\ldots\|ms_{i,1}$. Store each pair $(u_i,\ \boldsymbol{ms}_i)$ in a table U_i indexed by u_i. The final size of U_i is 2^{C-1}.
   (b) For each $\boldsymbol{ml}_i \in SL_i$, compute $v_i = h_2(y'_i, \overleftarrow{\boldsymbol{ml}}_i)$ where $\boldsymbol{ml}_i = ml_{i,1}\|ml_{i,2}\|\ldots\|ml_{i,s-1}\|ml_{i,s}$ and $\overleftarrow{\boldsymbol{ml}}_i = ml_{i,s}\|ml_{i,s-1}\|\ldots\|ml_{i,1}$. Where $s = C(2^{i-(C-1)-1}+1)$ if $C-1 < i \leq C-1+k$ and $s = C+i$ if $1 \leq i \leq C-1$. Store each pair $(v_i,\ \boldsymbol{ml}_i)$ in a table V_i indexed by v_i. The final size of V_i is 2^{C-1}.
   (c) Find a match $u_i = v_i$ between U_i and V_i, denote the matched state by $y'_{i-1} = u_i$. Combine the corresponding message fragment $\boldsymbol{ms}_i$ indexed by y'_i with MS and $\boldsymbol{ml}_i$ indexed by y'_i with ML, i.e. $MS = \boldsymbol{ms}_i\|MS$ and $ML = \boldsymbol{ml}_i\|ML$.

Then, for any length κ lying in the appropriate range of $[C(C-1)+kC,\ C^2-1+C(2^k+k-1)]$, one can construct a message M_e mapping $\hat{x}$ to $\tilde{y} = y'_0$ through h_1 and h_2 by picking messages fragment either from MS or from ML as described in [7]:

1. Select the length $\kappa' \in [C(C-1), C^2-1]$ such that $\kappa' = \kappa \bmod C$, defining the first $C-1$ message fragment choices: Selecting the message fragment $\boldsymbol{ms}_i$ in MS for $1 \leq i \leq C-1$ and $i \neq \kappa' - C$; Selecting the message fragment $\boldsymbol{ml}_i$ in ML for $i = \kappa' - C$.
2. Compute $kp \leftarrow (\kappa - \kappa')/C$ which is an integer in the range of $[k, 2^k + k - 1]$ and select the final k message fragment choices as in a standard expandable message using the binary representation of $kp - k$.

5.4 Complexity of Second Preimage Attack on Zipper Hash

The computational complexity of this second preimage attack on Zipper hash are summarized as follows, where the constant and polynomial factors are ignored.

- Step 1: $2^{\frac{n}{2}}$

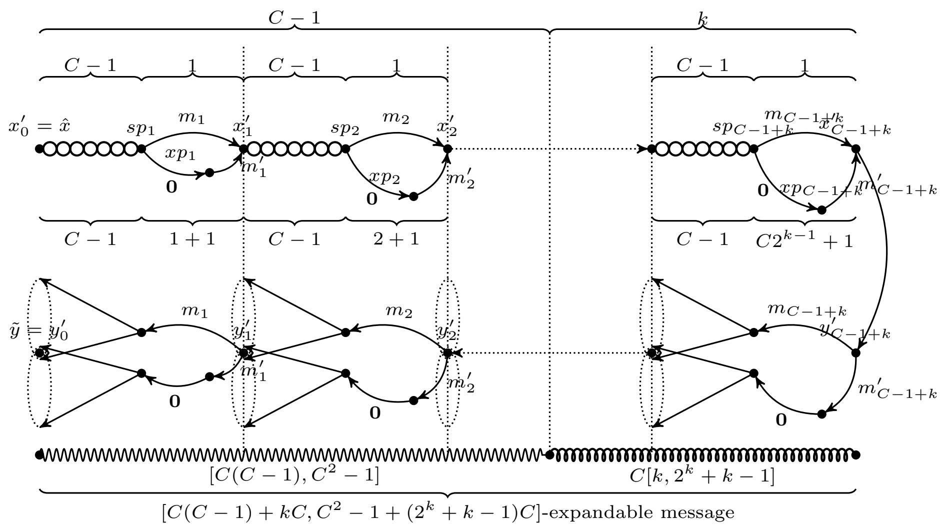

Fig. 10: Flowchart of constructing a simultaneous expandable message

- Step 2: $2 \cdot 2^t$
- Step 3: $r \cdot 2^{\frac{n}{2}}$
- Step 4: $n \cdot 2^k + n^2 \cdot 2^{\frac{n}{2}+\log_2(n)} = 2^{l'} + 2^{\frac{n}{2}+2\log_2(n)+1}$
- Step 5: 2^{n-l}
- Step 6: $2^r \cdot 2^{n-t}$
- Step 7: $2^r \cdot 2^{n-t}$
- Step 8: 2^r
- Step 9: $\mathcal{O}(2^{l'})$

According to Theorem 6, it is expected to sampling 2^{2n-3w} different starting point pairs until they simultaneously hit two α-nodes $(\bar{x},\ \bar{y})$ using up to 2^w iterates in two independent functional graphs of f_1 and f_2, where w is the allowed maximum distance to reach $\bar{x}$ and $\bar{y}$, which is determined by the allowed length L' of the second preimage, and is set to be $L' - q - 2r - C(C-1) - kC$. Thus, the number of messages in $\mathcal{M}_1$ which is set to be 2^r and that in $\mathcal{M}_2$ which is also set to be 2^r should satisfy $2^r \times 2^r = 2^{2n-3w} \approx 2^{2n-3l'}$, then $2^r \approx 2^{n-\frac{3}{2}l'}$.

The computational complexity is dominated by Step 2, 4, 5, 6, 7. In Step 2, we develop 2^t nodes for each functional graph of f_1 and f_2. So it requires $2 \cdot 2^t$ function calls. In Step 4, we build a simultaneous expandable message by a slightly modified constructing method proposed in [7]. The complexity is almost the same as that in [7]. We refer to [7] for the detailed discussion on the complexity of this step. Complexity of Step 5 depends on the probability of a collision $h_2(\bar{y}, \bar{m}) = y_q$ where y_q has $L = 2^l$ choices. According to the birthday paradox, it requires $2^n/L = 2^{n-l}$ trails for a collision. Complexity of Step 6 (resp. Step 7) depends on the number of starting point $\check{x}$ (resp. $\check{y}$) and the expected number of iterates before the chain $\boldsymbol{x}$ (resp. $\boldsymbol{y}$) hits G_1 (resp. G_2). Considering

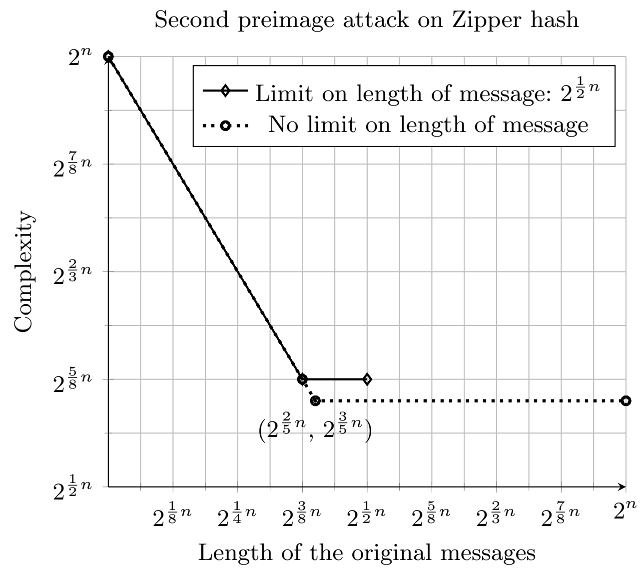

Fig. 11: Trade-offs between the message length and the complexity

that the size of G_1 (resp. G_2) is 2^t, it is expected to require 2^{n-t} iterates before a chain hits a point in G_1 starting from a random point. Thus, the computational complexity of Step 6 (resp. Step 7) is $r \cdot 2^r + 2^r \cdot 2^{n-t} = \Theta(2^r \cdot 2^{n-t})$. Complexity of Step 8 depends on numbers of entries in $\mathcal{T}_1$ and $\mathcal{T}_2$ which are upper bounded by the size of $\mathcal{M}_1$ and $\mathcal{M}_2$.

- When the allowed length L' of the second preimage is limited by $2^{\frac{n}{2}}$, then $2^w \approx 2^{l'}$ and $2^r = 2^{n-\frac{3}{2}w} \approx 2^{n-\frac{3}{2}l'}$. To achieve optimal complexity, we balance Step 2 and Step 6, 7 by setting $t = r + n - t \Rightarrow t = n - \frac{3}{4}l'$. We set l' to be the allowed maximum value, that is we set $2^{l'} = 2^{\frac{n}{2}} \Rightarrow 2^t = 2^{\frac{5}{8}n}$. This attack is valid and faster than 2^n as long as $l < n$.
- When the allowed length L' of the second preimage is not limited and can be greater than $2^{\frac{n}{2}}$, we utilize multi-cycles to reach $\bar{x}$ and $\bar{y}$ simultaneously as the technique used to improve preimage attack on XOR combiner in Section 4. In this case, we set $r = \frac{\frac{n}{2}-(l'-\frac{n}{2})}{2}$. That is because, the number of pairs $(\check{x}, \check{y})$ is 2^{2r}, and the required number of pairs to find one pair simultaneously reaching $(\bar{x}, \bar{y})$ is $2^{2n-\frac{3n}{2}}/2^{l'-\frac{n}{2}}$ when multi-cycles are used to amplify the probability of linking each pair to $(\bar{x}, \bar{y})$ at a common distance. Thus, we set $2r = 2n - \frac{3n}{2} - (l' - \frac{n}{2}) \Rightarrow r = \frac{\frac{n}{2}-(l'-\frac{n}{2})}{2}$.
  - For $\frac{3}{8}n < l \leq \frac{2}{5}n$, we can set $l' = n - l$. And balance Step 2 and Step 6, 7 by setting $t = r + n - t = \frac{\frac{n}{2}-(l'-\frac{n}{2})}{2} + n - t = \frac{l}{2} + n - t \Rightarrow t = \frac{l}{4} + \frac{n}{2}$. Since $l \leq \frac{2}{5}n$, one have $t = \frac{l}{4} + \frac{n}{2} \leq n - l$. Thus, complexity of Step 5, i.e. 2^{n-l}, is the dominating part.

- For $\frac{2}{5}n < l$, we set $l' = \frac{3}{5}n$ and limit $q < l' = \frac{3}{5}n$ where q is the merged point of the second message to the state chain of the original message. And balance Step 2 and Step 6, 7 by setting $t = \frac{\frac{n}{2}-(l'-\frac{n}{2})}{2} + n - t = \Rightarrow t = \frac{3}{5}n$. Thus, keep a stable complexity $2^{\frac{3}{5}n}$ for attack on messages of length $l > \frac{2}{5}n$.

A trade-off curve for these cases is shown in Fig. 11. In these attacks, length of the constructed second message L' is $2^{n/2}$ when limit on length of message is $2^{n/2}$, or $l' = n - l$ for $\frac{3}{8}n < l \leq \frac{2}{5}n$ and $l' = \frac{3}{5}n$ for $\frac{2}{5}n < l$ when no limit on length of message.

Remark 3. We notice that, when $l < \frac{2}{5}n$, the complexity of this second preimage attack on Zipper hash is dominated by Step 5. Thus, in this case the strength of Zipper Hash (with each pass using Merkle-Damgård-structure compression functions) against second preimage attack is no more than that of a single pass Merkle-Damgård-structure Hash function.

5.5 Experimental Results

We have simulated the entire process of this second preimage attack on Zipper hash (simulated h_1 with chopped AES-128 and h_2 with chopped SM4-128) for $n = 24$ and $n = 32$ with $t = \frac{5}{8}n$, $r = \frac{1}{4}n + 1$. In our simulations we preformed 1000 times attack for $n = 24$ and 100 times for $n = 32$. The success probability is 0.684 for $n = 24$ and is 0.8 for $n = 32$. The number of function calls for each step in those attacks are all as expected.

6 Conclusion

In this paper, we proposed the first second-preimage attack on Zipper hash and improved preimage attack on the XOR combiner with two narrow-pipe Merkle-Damgård hash functions. These attacks are based on functional graph of a random mapping. A future work might be to further investigate properties of functional graph in order to improve generic attacks on hash combiners, *e.g.*, reducing the complexity of generic attacks to match the lower security bounds, or shortening the length of (second) preimage messages.

Acknowledgments. Lei Wang and Dawu Gu are sponsored by National Natural Science Foundation of China (61602302, 61472250, 61672347), Natural Science Foundation of Shanghai (16ZR1416400), Shanghai Excellent Academic Leader Funds (16XD1401300). The authors would like to thank the anonymous reviewers of CRYPTO 2017 for their comments and suggestions.

References

1. Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.): Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, LNCS, vol. 5126. Springer (2008)
2. Allen, C., Dierks, T.: The TLS Protocol Version 1.0. RFC 2246 (Jan 1999), https://rfc-editor.org/rfc/rfc2246.txt
3. Andreeva, E., Bouillaguet, C., Dunkelman, O., Kelsey, J.: Herding, Second Preimage and Trojan Message Attacks beyond Merkle-Damgård. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) Selected Areas in Cryptography, SAC 2009. Revised Selected Papers. LNCS, vol. 5867, pp. 393–414. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
4. Brassard, G. (ed.): Advances in Cryptology - CRYPTO' 89, LNCS, vol. 435. Springer (1990)
5. Chen, S., Jin, C.: A Second Preimage Attack on Zipper Hash. Security and Communication Networks 8(16), 2860–2866 (2015)
6. Damgård, I.: A Design Principle for Hash Functions. In: Brassard [4], pp. 416–427
7. Dinur, I.: New Attacks on the Concatenation and XOR Hash Combiners. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology - EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 484–508. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
8. Dinur, I., Leurent, G.: Improved generic attacks against hash-based macs and haifa. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 149–168. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
9. Fischlin, M., Lehmann, A.: Security-Amplifying Combiners for Collision-Resistant Hash Functions. In: Menezes, A. (ed.) Advances in Cryptology - CRYPTO 2007. LNCS, vol. 4622, pp. 224–243. Springer (2007)
10. Fischlin, M., Lehmann, A.: Multi-Property Preserving Combiners for Hash Functions. In: Canetti, R. (ed.) Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008. LNCS, vol. 4948, pp. 375–392. Springer (2008)
11. Fischlin, M., Lehmann, A., Pietrzak, K.: Robust Multi-Property Combiners for Hash Functions Revisited. In: Aceto et al. [1], pp. 655–666
12. Fischlin, M., Lehmann, A., Pietrzak, K.: Robust Multi-Property Combiners for Hash Functions. Journal of Cryptology 27(3), 397–428 (2014)
13. Flajolet, P., Odlyzko, A.M.: Random Mapping Statistics. In: Quisquater, J.J., Vandewalle, J. (eds.) Advances in Cryptology - EUROCRYPT' 89. LNCS, vol. 434, pp. 329–354. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)
14. Freier, A.O., Karlton, P., Kocher, P.C.: The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (Aug 2011), https://rfc-editor.org/rfc/rfc6101.txt
15. Guo, J., Peyrin, T., Sasaki, Y., Wang, L.: Updates on Generic Attacks against HMAC and NMAC. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 131–148. Springer (2014)
16. Hellman, M.E.: A Cryptanalytic Time-Memory Trade-off. IEEE Trans. Information Theory 26(4), 401–406 (1980)
17. Hoch, J.J., Shamir, A.: Breaking the ICE - Finding Multicollisions in Iterated Concatenated and Expanded (ICE) Hash Functions. In: Robshaw, M.J.B. (ed.) Fast Software Encryption - FSE 2006, Revised Selected Papers. LNCS, vol. 4047, pp. 179–194. Springer (2006)

18. Hoch, J.J., Shamir, A.: On the Strength of the Concatenated Hash Combiner When All the Hash Functions Are Weak. In: Aceto et al. [1], pp. 616–630
19. Jha, A., Nandi, M.: Some Cryptanalytic Results on Zipper Hash and Concatenated Hash. Cryptology ePrint Archive, Report 2015/973 (2015), http://eprint.iacr.org/2015/973
20. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M. (ed.) Advances in Cryptology - CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
21. Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
22. Lehmann, A.: On the Security of Hash Function Combiners. Ph.D. thesis, Darmstadt University of Technology (2010)
23. Leurent, G., Peyrin, T., Wang, L.: New Generic Attacks against Hash-Based MACs. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology - ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 1–20. Springer (2013)
24. Leurent, G., Wang, L.: The Sum Can Be Weaker Than Each Part. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 345–367. Springer (2015)
25. Liskov, M.: Constructing an Ideal Hash Function from Weak Ideal Compression Functions. In: Biham, E., Youssef, A.M. (eds.) Selected Areas in Cryptography, SAC 2006, Revised Selected Papers. LNCS, vol. 4356, pp. 358–375. Springer (2006)
26. Mendel, F., Rechberger, C., Schläffer, M.: MD5 Is Weaker Than Weak: Attacks on Concatenated Combiners. In: Matsui, M. (ed.) Advances in Cryptology - ASIACRYPT 2009. LNCS, vol. 5912, pp. 144–161. Springer (2009)
27. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard [4], pp. 428–446
28. Nandi, M., Stinson, D.R.: Multicollision Attacks on Some Generalized Sequential Hash Functions. IEEE Trans. Information Theory 53(2), 759–767 (2007)
29. van Oorschot, P.C., Wiener, M.J.: Parallel Collision Search with Cryptanalytic Applications. J. Cryptology 12(1), 1–28 (1999)
30. Perrin, L., Khovratovich, D.: Collision Spectrum, Entropy Loss, T-Sponges, and Cryptanalysis of GLUON-64. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption - FSE 2014. Revised Selected Papers. LNCS, vol. 8540, pp. 82–103. Springer (2014)
31. Peyrin, T., Sasaki, Y., Wang, L.: Generic Related-Key Attacks for HMAC. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012. LNCS, vol. 7658, pp. 580–597. Springer (2012)
32. Peyrin, T., Wang, L.: Generic Universal Forgery Attack on Iterative Hash-Based MACs. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology - EUROCRYPT 2014. LNCS, vol. 8441, pp. 147–164. Springer (2014)
33. Peyrin, T., Wang, L.: Generic Universal Forgery Attack on Iterative Hash-Based MACs. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology - EUROCRYPT 2014. LNCS, vol. 8441, pp. 147–164. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
34. Rényi, A., Szekeres, G.: On the Height of Trees. Journal of the Australian Mathematical Society 7(4), 497–507 (11 1967)