Hindawi
IET Information Security
Volume 2024, Article ID 1230891, 7 pages
https://doi.org/10.1049/2024/1230891

Research Article

A Second Preimage Attack on the XOR Hash Combiner

Shiwei Chen, Ting Cui, Chenhui Jin, and Congjun Wang

The Department of Applied Mathematics, PLA SSF Information Engineering University, Zhengzhou 450004, China

Correspondence should be addressed to Shiwei Chen; chenshiwei1012@126.com

Received 24 October 2023; Revised 4 March 2024; Accepted 12 March 2024; Published 22 March 2024

Academic Editor: Tom Chen

The exclusive-or (XOR) hash combiner is a classical hash function combiner, which is well known as a good PRF and MAC combiner, and is used in practice in TLS versions 1.0 and 1.1. In this work, we analyze the second preimage resistance of the XOR combiner underlying two different narrow-pipe hash functions with weak ideal compression functions. To control simultaneously the behavior of the two different hash functions, we develop a new structure called multicollision-and-double-diamond. Multicollision-and-double-diamond structure is constructed using the idea of meet-in-the-middle technique, combined with Joux's multicollision and Chen's inverse-diamond structure. Then based on the multicollision-and-double-diamond structure, we present a second preimage attack on the XOR hash combiner with the time complexity of about $O((2n+1)2^{n/2}+(n-l)2^{n-l}+(n-k)2^{n-k}+2^{l+1}+2^{k+1})$ (n is the size of the XOR hash combiner and l and k are respectively the depths of the two inverse-diamond structures), less than the ideal time complexity $O(2^n)$, and memory of about $O(2^k+2^l)$.

1. Introduction

Hash functions are important cryptographic primitives in modern cryptography, mainly used in many cryptographic protocols, message authentications, etc. A hash function H is to transfer a message M with arbitrary length into a message digest h with fixed length called as a hash value. If the length of the hash value is n bits, we call the hash function as *n-bit hash function*. To guarantee the security of the applications, a n-bit hash function H needs to satisfy the following three basic security principles:

Preimage resistance: Given a n-bit hash value h, it should be difficult to find a message M such that $h=H(M)$.

Second preimage resistance: Given a message M and the corresponding n-bit hash value $h=H(M)$, it should be difficult to find another message M' such that $H(M')=h$.

Collision resistance: It should be difficult to find two different messages (M, M') such that $H(M')=H(M)$.

For a n-bit hash function, if the computational complexity of finding a second preimage or a preimage is less than 2^n, then the hash function is considered not to be second preimage resistance or preimage resistance.

Since the devastating attacks of Wang and Yu [1] on the MD family were proposed, to guarantee the security of the hash function a practical countermeasure might be using two different hash functions simultaneously in a combiner. The combined hash function is thought to be at least as security as any one of them. That is to say, it is only broken when both two hash functions were weak. There are two classical hash combiners, that is, concatenation hash combiner $H_1(M)||H_2(M)$ and XOR hash combiner $H_1(M)\oplus H_2(M)$. In [2], Joux used the multicollision to prove that finding a collision for the concatenation hash combiner with two narrow-pipe hash functions is not harder than finding a collision on one of the two hash functions. More precisely, though the concatenation hash combiner has $2n$ bits hash value, it only offers the security close to a n-bit hash function. The XOR hash combiner has the same length of hash value as the two hash functions. That is to say, the XOR hash combiner is length preserving, which increases the difficulty of analyzing this combiner. In particular, the designer of TLS [3] used the sum of HMAC-MD5 and HMAC-SHA-1 as the key derivation function and claimed that the combined hash function should guarantee its security if either algorithm remains secure. Moreover, Hoch and Shamir [4] proved that there are no generic attacks with complexity smaller than $O(2^{n/2})$, which is tight for collision resistance. However, for preimage attack or second preimage attack, there still exists a gap between $2^{n/2}$ and the expected bound 2^n. In Eurocrypt 2015, Leurent and Wang [5] developed a novel structure to control simultaneously the two independent hash functions with the same input message and thereby presented a generic preimage attack on the XOR combiner with two narrow-pipe hash functions with the complexity of $O(2^{5n/6})$ less than the ideal complexity of $O(2^n)$. Then, Dinur [6] improved the result in [5] and devised a new preimage attack on XOR hash combiner with two Merkle-Damgård (MD) hash functions with the complexity of $O(2^{2n/3})$. These two results reflect that the XOR combiner cannot offer the same security as an n-bit hash function. In CRYPTO 2017, Bao et al. [7] proposed an improved preimage attack against the XOR hash combiner based on the functional graph with a complexity of $O(2^{5n/8})$. Then, in 2020 Bao et al. [8] proposed several generic preimage attacks on the XOR hash combiner. Recently, Dong et al. [9] proposed a quantum preimage attack on the XOR hash combiner. However, there is no paper researching on the capability of the XOR hash combiner resisting the second preimage attack.

In this work, we research on the second preimage resistance of the XOR combiner underlying two different narrow-pipe hash functions with weak ideal compression function. To control simultaneously the behavior of the two different hash functions, we develop a new structure called multicollision-and-double-diamond. Multicollision-and-double-diamond structure is constructed using the idea of meet-in-the-middle technique, combined with Joux's multicollision and inverse-diamond structure. Then based on the Multicollision-and-two-diamond structure, we present the first second preimage attack on the XOR hash combiner with the time complexity of about $O((2n+1)2^{n/2}+(n-l)2^{n-l}+(n-k)2^{n-k}+2^{l+1}+2^{k+1}$; n is the size of the XOR hash combiner and l and k is respectively the depth of the two inverse-diamond structures), less than the ideal time complexity $O(2^n)$, and memory complexity of about $O(2^k+2^l)$.

After setting up some preliminaries in Section 2, we propose our second preimage attack on XOR hash combiner in Section 3 and analyze the complexity of our attack. Section 4 concludes our results and discuss the next work.

2. Preliminaries

2.1. Merkle-Damgård Structure. MD structure proposed by Merkle [10] and Damgård [11] is a typical domain extension for hash functions, used in many hash function standards, such as MD5, SHA-0, SHA-1, and SHA-2. For an input message M with arbitrary length, an initial value IV and a compression function $f:\{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^n$, a hash function with MD structure $\mathrm{MD}_f(\mathrm{IV}, M)$ is processed as follows:

Step 1: Padding the input message M into M' such that the length of M' is the multiple of m. Then, divide the padded message into t m-bit message blocks $M_1, M_2, \ldots, M_t$.

Step 2: Let $h_0 = \mathrm{IV}$. For $i = 1$ to t, do

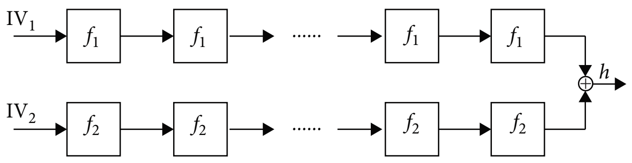

Figure 1: Schematic of XOR hash combiner.

$$h_i = f(h_{i-1}, M_i). \tag{1}$$

Step 3: Output h_t as the hash value of the input M.

If the length of the original message is padded in the last message block, the MD structure is called *the strengthened MD structure*, which is collision-resistance preserving.

2.2. Weak Ideal Compression Function. Since the hash function is onewayness, the compression function is not required invertible. However, to analyze the security of domain extension underlying a weak compression function, the weak ideal compression function was defined in [12], which is invertible.

Definition 1. (Weak ideal compression function) [12]. Let $f:\{0,1\}^{m+n} \rightarrow \{0,1\}^n$ be an ideal compression function. If for any input (x,z), the adversary can find a random value y such that $f(x,y)=z$, or for any input (y,z), the adversary can find a random value x such that $f(x,y)=z$, then we call f the weak ideal compression function.

2.3. Description of the XOR Hash Combiner. The XOR hash combiner utilizes two different hash functions H_1 and H_2, which produces the XOR sum of the hash values of two hash functions as the hash value of the XOR hash combiner (Figure 1). In this paper, we assume the two underlying hash functions are both MD structure hash functions.

For an input message M, we firstly pad it into M_{pad} with the length of multiple of m and divide it into p m-blocks $m_1, m_2, \ldots, m_p$. Then, for the two initial chaining values $\mathrm{IV}_1, \mathrm{IV}_2$ and two compression function $f_1, f_2:\{0,1\}^{m+n} \rightarrow \{0,1\}^n$, the XOR hash combiner $\mathrm{XOR\_Hash}_{\mathrm{MD}_{f_1},\mathrm{MD}_{f_2}}(\mathrm{IV}_1, \mathrm{IV}_2, M)$ is processed as follows:

Step 1. In the first pass, for the initial value IV_1 and the underlying compression function f_1, perform $\mathrm{MD}_{f_1}(\mathrm{IV}_1, M)$ as described in Section 2.1.

Step 2. In the second pass, for the initial value IV_2 and the underlying compression function f_2, perform $\mathrm{MD}_{f_2}(\mathrm{IV}_2, M)$ as described in Section 2.1.

Step 3. Compute the sum of the results obtained in Step 1 and Step 2 and then output it as the hash value of XOR hash combiner, that is,

$$\begin{aligned} &\mathrm{XOR\_Hash}_{\mathrm{MD}_{f_1},\mathrm{MD}_{f_2}}(\mathrm{IV}_1, \mathrm{IV}_2, M) \\ &= \mathrm{MD}_{f_1}(\mathrm{IV}_1, M) \oplus \mathrm{MD}_{f_2}(\mathrm{IV}_2, M) \end{aligned}. \tag{2}$$

In the following paper, we assume that the two compression functions f_1, f_2 in XOR hash combiner are both weak ideal compression functions.

2.4. Notations

H_1, H_2: two different hash functions with strengthened MD structure;

XOR_Hash$_{H_1,H_2}$: XOR hash combiner based on hash functions H_1, H_2;

f_1,f_2: two weak ideal compression function used in H_1 and H_2, respectively;

f_1^*,f_2^*: two hash functions with MD structure and compression function f_1 and f_2, respectively;

$\mathrm{IV}_1, \mathrm{IV}_2$: initial chaining values of hash functions H_1 and H_2, respectively;

M: input message with arbitrary length;

M_{pad}: padded message with the last block including padding;

m_i: the i-th message block of M_{pad};

$h_1^{(i)}, h_2^{(i)}$: the i-th chaining values produced by H_1 and H_2, respectively;

h: the n-bit hash values of XOR_Hash$_{H_1,H_2}$.

2.5. Existing Attack Techniques

2.5.1. Birthday Attack. Birthday attack is one of the best-known combinatorial tools in cryptology, and the birthday problem is described as follows.

Definition 2. (Birthday problem) [13]. Given two lists L_1, L_2 of elements drawn uniformly and independently at random from $\{0, 1\}^n$, find $x_1 \in L_1$ and $x_2 \in L_2$ such that $x_1 \oplus x_2 = 0$.

If the sizes of lists are favorably chosen, the complexity of the optimal algorithm is about $O(2^{n/2})$. Hence, whatever the lists L_1, L_2 are produced by one hash function or two different ones, we could find $x_1 \in L_1$ and $x_2 \in L_2$ such that $x_1 \oplus x_2 = 0$ with the complexity of $O(2^{n/2})$. That is, assuming H_1, H_2 are two different hash functions, then we could find two different inputs (chaining values or messages) I_1, I_2 such that $H_1(I_1) \oplus H_2(I_2) = 0$ with the time complexity of $O(2^{n/2})$. Furthermore, for any fixed values $a \in \{0, 1\}^n$, we also could find two different inputs (chaining values or messages) I_1, I_2 such that $H_1(I_1) \oplus H_2(I_2) = a$ with the time complexity of $O(2^{n/2})$.

2.5.2. Joux's Multicollision. A 2^k-collision attack is to find 2^k messages producing the same hash value. That is, for a n-bit hash function H and t l-block messages $M_1, M_2, \ldots, M_t$, if

$$H(M_1) = H(M_2) = \cdots = H(M_t), \tag{3}$$

then we call $M_1, M_2, \ldots, M_t$ as *t-collision* with length of l-block of the hash function H.

From [14], we know that the time complexity of finding a 2^k-collision is about $O(2^{(2^k-1)n/2^k})$ for a n-bit ideal hash function.

In 2004, Joux [2] proposed a multicollision attack on hash function H with MD construction and an ideal compression function f. Assume that the initial value is IV, and set $h_0 = \mathrm{IV}$. Then Joux's multicollision attack is processed as follows to find one 2^k-collision (Figure 2).

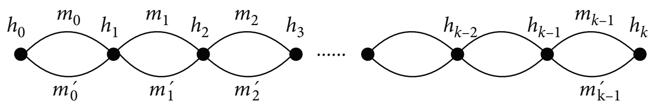

FIGURE 2: Construction of Joux's multicollision.

Step 1. Use the birthday attack to find two one-block messages m_0, m_0' such that

$$f(h_0, m_0) = f(h_0, m_0') = h_1 \ . \tag{4}$$

Step 2. From the chaining value h_1, find two one-block messages m_1, m_1' such that

$$f(h_1, m_1) = f(h_1, m_1') = h_2. \tag{5}$$

Step 3. Repeat the above step, until find the $k-th$ two one-block messages m_{k-1}, m_{k-1}' such that

$$f(h_{k-1}, m_{k-1}) = f(h_{k-1}, m_{k-1}') = h_k \ . \tag{6}$$

Step 4. For each i with $0 \le i \le k-1$, choose m_i or m_i' to form 2^k messages with the length of k-block producing the same hash value h_k.

From the above description of birthday attack, we know that the time complexity of finding a 2-collision is about $O(2^{n/2})$, so for any i with $0 \le i \le k-1$, the time complexity of finding the i-th two one-block messages m_i, m_i' is about $O(2^{n/2})$. Therefore, the time complexity of the above multicollision attack is about $O(k \cdot 2^{n/2})$.

2.5.3. Diamond Structure. A k-depth diamond structure proposed by Kelsey and Kohno [15] starts from 2^k different random values and ends with one point, which is essentially a 2^k-multicollision. And the complexity of constructing a k-depth diamond multicollision is about $O(2^{n/2+k/2})$ where the birthday attack is mainly applied. Since the 2^k different starting points in the diamond structure offer greater choose space, it is not only useful for the single-pass hash function, but also for the multi-pass hash function, such as the preimage attack on concatenation and XOR hash combiners [6], the second preimage attack on zipper hash [16], and so on.

2.5.4. Inverse-Diamond Structure. A l-depth inverse-diamond structure (Figure 3) proposed in [16] is to produce many ended points from one starting point, and the messages used to produce different chaining values are from the multicollision constructed in another pass, of which the time complexity is about $O(l2^{n/2} + 2^{l+1})$. Chen and Jin [16] presented the first second preimage attack on zipper hash using the inverse-diamond structure, with the time complexity of $O((2k+n)2^{n/2} + 2^{n-k} + (n-l)2^{n-l} + 2^{l+1})$, less than $O(2^n)$. The details of constructing an inverse-diamond structure are described as follows.

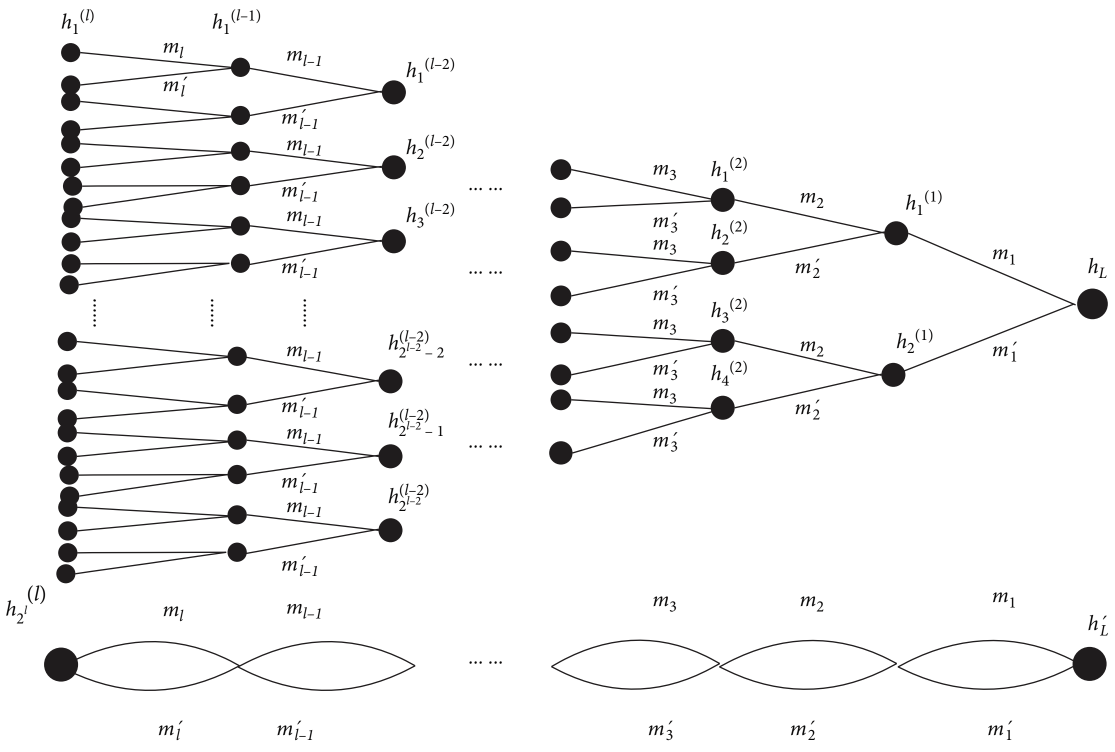

Figure 3: Construction of the inverse-diamond structure.

Step 1. In the second pass, from the starting point h_L', construct a 2^l-collision with the length of l-block on the weak ideal compression function f_2 according to Joux's method. For any i with $1 \le i \le l$, denote the two messages produced in i-th step as m_i, m_i', respectively.

Step 2. In the first pass, from the starting point h_L, compute $f_1^{-1}(h_L, m_1)$ and $f_1^{-1}(h_L, m_1')$, which are denoted as $h_1^{(1)}$ and $h_2^{(1)}$, respectively.

Step 3. From the point $h_1^{(1)}$, compute out

$$f_1^{-1}\left(h_1^{(1)}, m_2\right) = h_1^{(2)}, f_1^{-1}\left(h_1^{(1)}, m_2'\right) = h_2^{(2)}. \quad (7)$$

Similarly, from the $h_2^{(1)}$, compute out

$$f_1^{-1}\left(h_2^{(1)}, m_2\right) = h_3^{(2)}, f_1^{-1}\left(h_2^{(1)}, m_2'\right) = h_4^{(2)}. \quad (8)$$

Step 4. Repeatedly compute from the points obtained in the last step until the l-th step. That is, for any i with $1 \le i \le 2^{l-1}$, from $h_i^{(l-1)}$, compute out

$$f_1^{-1}\left(h_i^{(l-1)}, m_l\right) = h_{2i-1}^{(l)}, f_1^{-1}\left(h_i^{(l-1)}, m_l'\right) = h_{2i}^{(l)}. \quad (9)$$

Therefore, we obtain 2^l points at the end.

From the constructing of the inverse-diamond structure, we know that it could guarantee the messages in two passes identical, which is important for analyzing the two-pass hash combiners.

3. Our Second Preimage Attack on XOR Hash Combiner

The XOR hash combiner is the sum of two different n-bit hash functions with the same input message and different initial chaining values, so to find the second preimage of the target hash value, the key problem is how to keep the processing of the second preimage in two passes consistent. To solve this problem, we develop a new technique called *multi-collision-and-double-diamond structure*, and then propose a second preimage attack on the XOR hash combiner with the time complexity of $O((2n+1)2^{n/2} + 2^{l+1} + (n-l)2^{n-l} + 2^{k+1} + (n-k)2^{n-k})$, much less than the ideal time complexity $O(2^n)$.

3.1. Constructing of the Multicollision-and-Double-Diamond Structure. Based on the inverse-diamond structure, we bring in the idea of meet-in-the-middle technique and thereby propose an extended inverse-diamond structure called *multi-collision-and-double-diamond* structure. The *multicollision-and-double-diamond* structure is to keep the messages in two passes consistent, mainly including one multicollision structure and two inverse-diamond structures. In the (n,k)-*multi-collision-and-double-diamond* structure, we firstly construct a multicollision with the length of n in one pass using Joux's method. And then in another pass, to reduce the complexity of finding one second preimage consistent with the one in

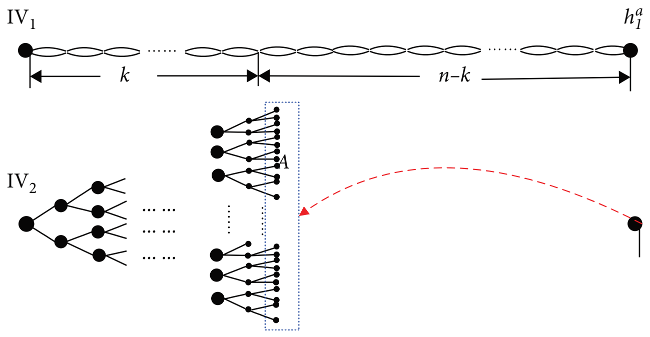

Figure 4: Construction of the multicollision-and-double-diamond structure.

the first pass, we construct two inverse-diamond structures from two different starting points and utilize the idea of meet-in-the-middle technique to make them collide.

Now, assume that the two hash functions used in the XOR hash combiner are f_1^* and f_2^*. The details of constructing a (n, k)-*multicollision-and-double-diamond* structure for (f_1^*, f_2^*) is described as follows (Figure 4).

Step 1. In the first pass, from the initial chaining value IV_1 construct a 2^n-collision with length n using Joux's method and produce a chaining value h_1^a.

Step 2. In the second pass, from the initial chaining value IV_2 construct an inverse-diamond structure of hash function f_2^* with the depth of k producing 2^k points noted set A. And then from another given point $h_2^{(1)}$, construct an inverse-diamond structure of hash function $(f_2^{-1})^*$ with the depth of $(n-k)$ producing 2^{n-k} points noted set B, in which there is at least one point equal to one of the points in A since the size of the chaining value is n-bit and the probability is about 2^{-n}.

From the above description, we know that the multicollision-and-double-diamond structure could be constructed only if the values of two starting points in two passes and one of ending point are known in advance and the compression functions are invertible.

3.2. Our Second Preimage Attack on XOR Hash Combiner. To guarantee that the values of two starting points in two passes and one of ending point are known in advance, we firstly construct two muticollisons with length of n in two passes respectively. Then using the multicollision-and-double-diamond structure twice with different starting points, we present a first second preimage attack on the XOR hash combiner, of which the time complexity is about $O((2n+1)2^{n/2} + 2^{l+1} + (n-l)2^{n-l} + 2^{k+1} + (n-k)2^{n-k})$ and memory complexity is about $O(2^k + 2^l)$.

For a given hash value h and a padded message

$$M = m_1||m_2||\cdots||m_{t-1}||m_t\ , \tag{10}$$

where $m_i (1 \le i \le t)$ is n-bit message block and $m_t = w||\text{pad}$, we obtain a second preimage of the XOR hash combiner $\text{XOR\_Hash}_{\text{MD}_{f_1}, \text{MD}_{f_2}}(\text{IV}_1, \text{IV}_2, M)$ using the following procedure (Figure 5).

Step 1. Using the birthday attack to find two different chaining values PX and PY such that

$$f_1(PX, w||\text{pad}) \oplus f_2(PY, w||\text{pad}) = h\ , \tag{11}$$

where h is the target hash value and $w||\text{pad}$ is the last message block.

Step 2. Constructing a (n, k)-multicollision-and-double-diamond structure for (f_1^*, f_2^*) and (n, l)-multicollision-and-double-diamond structure for $((f_2^{-1})^*, (f_1^{-1})^*)$ simultaneously:

Step 2.1. In the first pass, from initial value IV_1 construct a 2^n-collision with length n for hash function f_1^* using Joux's multicollision method and produce a chaining value h_1^a.

Step 2.2. In the second pass, from the chaining value PY, construct a 2^n-collision with length n for the hash function $(f_2^{-1})^*$ using Joux's multicollision method and the end point is noted h_2^b.

Step 2.3. In the first pass, from the chaining value PX, construct an inverse-diamond construction with depth l for the hash function $(f_1^{-1})^*$ and the set of points is noted A_1 of which the size is 2^l.

Step 2.4. Find a message M_3 with length of $n-l$ from the 2^{n-l}-collision constructed in the first pass such that $f_1^*(h_1^a, M_3)$ equals to one of the points in set A_1, which leads a message M_4 with length of l-block in the inverse-diamond producing the chaining value PX.

Step 2.5. In the second pass, from the initial chaining value IV_2, construct an inverse-diamond with depth of k using the 2^k-collision in the first pass and the set of end points is noted B. Find a message M_2 with length $n-k$ from the 2^{n-k}-collision constructed in the first pass such that $(f_2^{-1})^*(h_2^b, M_2)$ equals to one of the value in set B_1 and the corresponding message is noted as M_1 satisfying

$$f_2^*(\text{IV}_2, M_1) = (f_2^{-1})^*(h_2^b, M_2). \tag{12}$$

Step 3. Output the message

$$M_1||M_2||M_3||M_4||w||\text{pad}\ , \tag{13}$$

as the second preimage of the given hash value h.

3.3. Complexity of Our Second Preimage Attack. In this section, we analyze the time complexity and memory of our attack.

In Step 1, we find PX and PY using the birthday attack and since the hash size is n-bit, the time complexity is about $O(2^{n/2})$. According to Joux's multicollision method, the time complexity of Step 2.1 and Step 2.2 are both about $O(n2^{n/2})$. In Step 2.3, the time complexity of constructing an inverse-diamond structure with depth l is about $O(2^{l+1})$, and we need to store the 2^l values of set A_1. In Step 2.4, we compute 2^{n-l} messages with length of $n-l$ to collide with one of the values in set A_1, hence the time complexity is about $O((n-l)2^{n-l})$. And similarly, the time complexity of Step 2.5 is about $O((n-k)2^{n-k} + 2^{k+1})$ and the memory is about $O(2^k)$. In a word, the time complexity of our attack is about

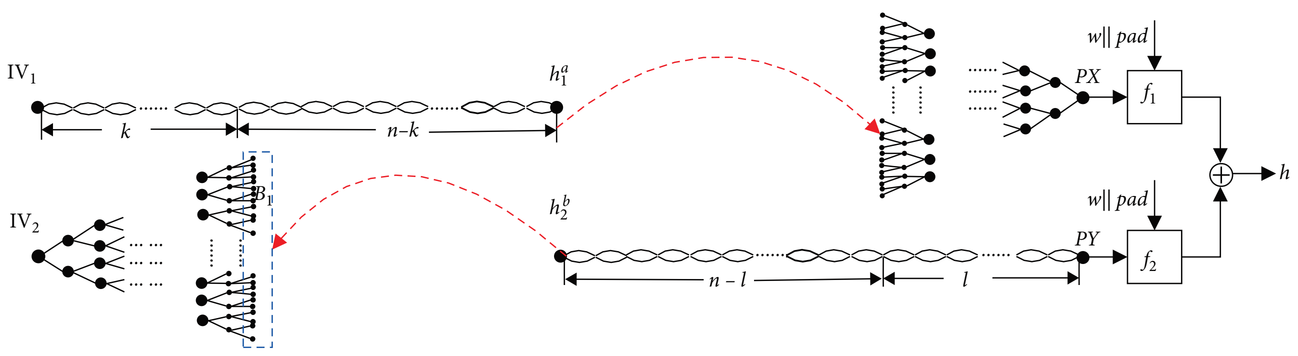

Figure 5: Scheme of our second preimage attack.

$$O\left((2n+1)2^{n/2}+2^{l+1}+(n-l)2^{n-l}+2^{k+1}+(n-k)2^{n-k}\right), \quad (14)$$

and the memory is about $O(2^k+2^l)$.

Specially, if $k=l=n/2$, then the time complexity is about $O(n2^{n/2})$ and the memory is about $O(2^{n/2})$.

4. Conclusion

In this work, we analyze the second preimage resistance of the XOR hash combiner underlying two different narrow-pipe hash functions with weak ideal compression function. To control simultaneously the behavior of the two different hash functions, we develop a new structure called *multicollision-and-double-diamond*, based on which we present a first second preimage attack on the XOR hash combiner with the time complexity of about $O((2n+1)2^{n/2}+2^{l+1}+(n-l)2^{n-l}+2^{k+1}+(n-k)2^{n-k})$ (n is the size of the XOR hash combiner and l and k are respectively the depths of the two inverse-diamond structures), less than the ideal time complexity $O(2^n)$, and memory complexity of about $O(2^k+2^l)$. Specially, if the depths of the two inverse-diamond structures are equal, then the time complexity is about $O(n2^{n/2})$ and the memory is about $O(2^{n/2})$. In the future work, we would like to analyze the security of the XOR hash combiner resisting other attacks.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

We are grateful to the anonymous referees for their valuable comments. The work in this paper is supported by the National Natural Science Foundation of China (Grant No: 62372463) and the Natural Science Foundation of Henan Province (Grant No: 222300420100).

References

[1] X. Wang and H. Yu, "How to break MD5 and other hash functions," in *Advances in Cryptology—EUROCRYPT 2005. EUROCRYPT 2005. Lecture Notes in Computer Science*, vol. 3494, pp. 19–35, Springer, Berlin, Heidelberg, 2005.

[2] A. Joux, "Multicollisions in iterated hash functions application to cascaded constructions," in *Advances in Cryptology–CRYPTO 2004. CRYPTO 2004. Lecture Notes in Computer Science*, vol. 3152, pp. 306–316, Springer, Berlin, Heidelberg, 2004.

[3] T. Dierks and C. Allen, *RFC2246: The TLS Protocol Version 1.0*, RFC Editor, United States, 1999.

[4] J. J. Hoch and A. Shamir, "On the strength of the concatenated hash combiner when all the hash functions are weak," in *Automata, Languages and Programming. ICALP 2008. Lecture Notes in Computer Science*, vol. 5126, pp. 616–630, Springer, Berlin, Heidelberg, 2008.

[5] G. Leurent and L. Wang, "The sum can be weaker than each part," in *Advances in Cryptology—EUROCRYPT 2015. EUROCRYPT 2015. Lecture Notes in Computer Science*, vol. 9056, pp. 345–367, Springer, Berlin, Heidelberg, 2015.

[6] I. Dinur, "New attacks on the concatenation and XOR hash combiners," in *Proceedings, Part I, of the 35th Annual International Conference on Advances in Cryptology—EUROCRYPT 2016–Vol-Volume 9665*, pp. 484–508, Springer-Verlag, Berlin, Heidelberg, May 2016.

[7] Z. Bao, L. Wang, J. Guo, and D. Gu, "Functional graph revisited: updates on (second) preimage attacks on hash combiners," in *Advances in Cryptology–CRYPTO 2017. CRYPTO 2017. Lecture Notes in Computer Science*, J. Katz and H. Shacham, Eds., vol. 10402, pp. 404–427, Springer, Cham, 2017.

[8] Z. Bao, I. Dinur, J. Guo, G. Leurent, and L. Wang, "Generic attacks on hash combiners," *Journal of Cryptology*, vol. 33, pp. 742–823, 2020.

[9] X. Dong, S. Li, P. Pham, and G. Zhang, "Quantum attacks on hash constructions with low quantum random access memory," in *Advances in Cryptology—ASIACRYPT 2023. ASIACRYPT 2023. Lecture Notes in Computer Science*, vol. 14440, pp. 3–33, Springer, Singapore, 2023.

[10] R. C. Merkle, "A certified digital signature," in *Advances in Cryptology—CRYPTO' 89 Proceedings. CRYPTO 1989. Lecture Notes in Computer Science*, vol. 435, pp. 218–238, Springer, New York, NY, 1990.

[11] I. B. Damgård, "A design principle for hash functions," in *Advances in Cryptology—CRYPTO' 89 Proceedings. CRYPTO 1989. Lecture Notes in Computer Science*, vol. 435, pp. 416–427, Springer New York, New York, NY, 1990.

[12] M. Liskov, “Constructing an ideal hash function from weak ideal compression functions,” in *Selected Areas in Cryptography. SAC 2006. Lecture Notes in Computer Science*, vol. 4356, pp. 358–375, Springer, Berlin, Heidelberg, 2006.

[13] D. Wagner, “A generalized birthday problem,” in *Advances in Cryptology—CRYPTO 2002. CRYPTO 2002. Lecture Notes in Computer Science*, vol. 2442, pp. 288–304, Springer, Berlin, Heidelberg, 2002.

[14] M. Nandi and D. R. Stinson, “Multicollision attacks on some generalized sequential hash functions,” *IEEE Transactions on Information Theory*, vol. 53, no. 2, pp. 759–767, 2007.

[15] J. Kelsey and T. Kohno, “Herding hash functions and the nostradamus attack,” in *Advances in Cryptology–EUROCRYPT 2006. EUROCRYPT 2006. Lecture Notes in Computer Science*, vol. 4004, pp. 183–200, Springer, Berlin, Heidelberg, 2006.

[16] S. Chen and C. Jin, “A second preimage attack on zipper hash,” *Security and Communication Networks*, vol. 8, no. 16, pp. 2860–2866, 2015.