New Distinguishing Attack on MAC using Secret-Prefix Method*

Xiaoyun Wang[1,2], Wei Wang[2], Keting Jia[2], and Meiqin Wang[2]

[1] Center for Advanced Study, Tsinghua University, Beijing 100084, China
xiaoyunwang@mail.tsinghua.edu.cn
[2] Key Laboratory of Cryptographic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China
wwang@math.sdu.edu.cn, kejia@mail.sdu.edu.cn, mqwang@sdu.edu.cn

Abstract. This paper presents a new distinguisher which can be applied to secret-prefix MACs with the message length prepended to the message before hashing. The new distinguisher makes use of a special truncated differential path with high probability to distinguish an inner near-collision in the first round. Once the inner near-collision is detected, we can recognize an instantiated MAC from a MAC with a random function. The complexity for distinguishing the MAC with 43-step reduced SHA-1 is $2^{124.5}$ queries. For the MAC with 61-step SHA-1, the complexity is $2^{154.5}$ queries. The success probability is 0.70 for both.

Key words: MAC, secret prefix method, distinguishing attack, SHA-1

1 Introduction

Message Authentication Code (MAC) algorithms play an important role in internet security protocols (SSL/TLS, SSH, IPsec) and the financial sector for debit and credit transaction. A MAC algorithm is a hash function with a secret key K as the secondary input, which guarantees data integrity and authenticity. The *secret prefix* method is a MAC construction which prepends a secret K to the message before the hashing operation, which is the basic design unit for HMAC/NMAC [1]. One suggestion to guarantee a secure secret prefix MAC is to prepend the message length to the message before hashing [13]. Recent work [2,3,15,16,17,19] discovered devastating collision attacks on hash functions from the MDx family. Such attacks have undermined the confidence in the most popular hash functions such as MD5 and SHA-1, and promoted the reevaluation of the actual security of the MAC algorithms based on them [4,6,8,11,12,14,18].

There are two kinds of distinguishing attacks on MACs, and they are respectively called *distinguishing-R* and *distinguishing-H* attacks [8]. Distinguishing-R attack means distinguishing a MAC from a random function, and distinguishing-H attack detects an instantiated MAC (by an underlying hash function or block

* Supported by the National Natural Science Foundation of China (NSFC Grant No. 60525201 and No.90604036) and 973 Project (No.2007CB807902).

cipher) from a MAC with a random function. Preneel and van Oorschot [10] introduced a general distinguishing-R attack on all iterated MACs using the birthday paradox, which requires about $2^{n/2}$ messages and works with a success rate 0.63, where n is the length of the hash output. Their attack can immediately be converted into a general forgery attack. For the distinguishing-H attack, its ideal complexity should be exhaustive search cost.

This paper focuses on the distinguishing-H attack that checks which cryptographic hash function is embedded in a MAC. For simplicity, we call it distinguishing attack.

Kim *et al.* [8] described two kinds of distinguishers for the HMAC structure, which are differential and rectangle distinguishers. For the differential distinguisher, it needs a collision differential path with probability higher than 2^{-n}, and the rectangle distinguisher needs a near-collision differential with probability higher than $2^{-n/2}$. For MD4, because it is easy to find a differential path with high probability [15,20], there are some successful cryptanalytic results on MACs based on MD4 [4,6,14]. For HMAC/NMAC-MD5, there is only one available differential path that is called dBB pseudo-collision path [5]. Because the dBB pseudo-collision consists of two different IVs and the same message, so all the attacks [4,6,11,14] are in the related-key setting. For HMAC/NMAC-SHA-0, there exists a partial key-recovery attack [4]. For HMAC-SHA-1, Kim *et al.* proposed a distinguishing attack on 43-step HMAC-SHA-1 with data complexity $2^{154.9}$, which was improved by Rechberger and Rijmen [12]. The improved attack detected 50-step HMAC-SHA-1 with data complexity $2^{153.5}$. The paper [12] also proposed a related-key distinguishing attack on 62-step (17-78) HMAC-SHA-1 and a full key-recovery attack on 34-step NMAC-SHA-1 in the related-key setting.

All the above attacks make use of collision or near-collision differential paths for the underlying compression function with probability higher than 2^{-n}. For MD5 and SHA-1, there are too many sufficient conditions in the collision or near-collision differential paths, which imply a complexity more than 2^n. Because most conditions focus on the first round, it is hard to analyze the MACs with MD5/SHA-1 for more steps without related keys. In this paper, we only consider the MACs with reduced SHA-1 starting from the first step.

One recent work [18] presented a new distinguishing attack on HMAC/NMAC-MD5 and MD5-MAC without related keys. Their distinguisher detects an inner near-collision in the first iteration. This motivates us to explore a similar attack on MACs based on SHA-1. For the MAC with SHA-1, the situation is more complex, because SHA-1 dose not have a differential path with high probability. If we do not consider the first round, the probability of the existing differential paths for the last three rounds is high. How to avoid the differential path in the first round, and completely explore the probability advantage in the last three rounds? We neglect the exact path in the first round, replace it with an inner near-collision, and explore the new techniques to detect the inner near-collision by a birthday attack. Our new distinguishing attack is applicable to secret-prefix MACs with the length prepended before hashing, which is denoted as LPMAC.

For LPMAC based on 43-step SHA-1, the complexity is $2^{124.5}$ queries, and for LPMAC with 61-step SHA-1, the complexity is $2^{154.5}$.

This paper is organized as follows: Section 2 gives brief descriptions of LPMAC and SHA-1. In Section 3, we present the new distinguisher which is available to LPMAC structure, and describe the details of the distinguishing attack on LPMAC with 61-step SHA-1 in Section 4. Finally, we conclude the paper in Section 5.

2 Backgrounds and Definitions

In this section, we define the notations used in this paper, and give brief descriptions of the LPMAC and SHA-1.

2.1 Notations

H	:	a hash function
$\overline{H}$	:	a hash function without padding and length appending
n	:	the length of the hash output
b	:	the length of one message block
IV	:	the initial chaining value
$x\|\|y$	:	the concatenation of the two bitstrings x and y
$x_{i,j}$	:	the j-th bit of x_i, where x_i is a 32-bit word, $j = 1, \ldots, 32$, and 32 is the most significant bit
$+, -$	:	addition and subtration modular 2^{32}
$\Delta^{-}x$	:	modular difference $x - x'$, where x and x' are two 32-bit words
$\wedge, \neg, \vee, \oplus$	:	bitwise AND, NOT, OR and exclusive OR
$\lll s$	:	left-rotation operation by s-bit

2.2 MAC using Secret Prefix Method

The *secret prefix* method is to append a message M to a secret key K before the hashing operation:

$$\text{Secret-Prefix-MAC}_K(M) = H(K\|M),$$

where H is an unkeyed hash function. This method was proposed in the 1980s, and suggested for MD4 independently in [7,13]. The original secret prefix MAC is insecure: given a message and its MAC, an attacker can easily append another message to the message and update the MAC accordingly, as the given MAC value can be taken as the initial chaining value for the appended message [10].

Prefixing the message length to the message before hashing is one suggestion to avoid the above attack [13]. However, our new distinguisher specifically works for this kind of MAC, which we call LPMAC. We provide that $K\|\#length\|pad$ is a full block, and this kind of MAC corresponds to a hash function $\overline{H}$ with a secret IV (K'), which is denoted as:

$$LPMAC_K(M) = \overline{H}(K\|\#length\|pad\|M) = \overline{H}_{K'}(M).$$

In the rest of this paper, LPMAC refers to a MAC with the new form $\overline{H}_{K'}(M)$.

2.3 Brief Description of SHA-1

The hash function SHA-1 was issued by NIST in 1995 as a Federal Information Processing Standard [9]. It follows the Merkle-Damgård iterative construction, takes a message M with the bit-length less than 2^{64}, and produces a 160-bit digest. The compression function takes a 160-bit chaining value $h^i = (a_0, b_0, c_0, d_0, e_0)$ and a 512-bit message block M^i as inputs, and produces another 160-bit chaining value h^{i+1}, where h^0 is the initial value IV, and $M = M^0 \| \cdots \| M^{t-1}$. By iterating all the message blocks M^i, we obtain the final 160-bit value h^t which is the hash value.

Each 512-bit block M^i is divided into sixteen 32-bit words, which is denoted as $(m_0, m_1, \ldots, m_{15})$. The message words are expanded to eighty 32-bit words $w_0, \ldots, w_{79}$:

$$w_j = \begin{cases} m_j, & \text{if } j = 0, \ldots, 15, \\ (w_{j-3} \oplus w_{j-8} \oplus w_{j-14} \oplus w_{j-16}) \lll 1, & \text{if } j = 16, \ldots, 79. \end{cases}$$

The compression function consists of 4 rounds, and each round includes 20 steps. The details for the compression function are the following:

- Input: $w_0, \ldots, w_{79}$ and $h^i = (a_0, b_0, c_0, d_0, e_0)$, where h^i is a 160-bit chaining value .
- Step update: For $j = 1, \ldots, 80$,

$$a_j = (a_{j-1} \lll 5) + f_j(b_{j-1}, c_{j-1}, d_{j-1}) + e_{j-1} + w_{j-1} + k_j,$$
$$b_j = a_{j-1},\ c_j = b_{j-1} \lll 30,\ d_j = c_{j-1},\ e_j = d_{j-1},$$

  where the Boolean function f_j and constant k_j are described in Table 1.
- Output: $h^{i+1} = (a_0 + a_{80}, b_0 + b_{80}, c_0 + c_{80}, d_0 + d_{80}, e_0 + e_{80})$.

Table 1. Boolean Functions and Constants Involved in SHA-1

round	steps	f_j	k_j
1	1-20	IF : $(x \wedge y) \vee (\neg x \wedge z)$	0x5a827999
2	21-40	XOR : $x \oplus y \oplus z$	0x6ed6eba1
3	41-60	MAJ : $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$	0x8fabbcdc
4	61-80	XOR : $x \oplus y \oplus z$	0xca62c1d6

3 New Distinguisher on LPMAC Structure

This section introduces a new distinguisher of the LPMAC structure. It is based on a near-collision differential path with two message blocks.

3.1 Recent Attack on HMAC/NMAC-MD5 and MD5-MAC [18]

We first recall the distinguishing attack on HMAC/NMAC-MD5 and MD5-MAC presented in [18]. This distinguisher utilizes the dBB pseudo-collision path of MD5 [5], where the hash values collide with probability 2^{-46} when the IV difference satisfies the dBB-condition, i.e.,

$$IV \oplus IV' = (0x80000000, 0x80000000, 0x80000000, 0x80000000),$$
$$\mathrm{MSB}(B_0) = \mathrm{MSB}(C_0) = \mathrm{MSB}(D_0),$$

where $(A_0, B_0, C_0, D_0) = IV$, and MSB means the most significant bit. To discard the related-key setting, and give a real distinguishing attack, the adversary firstly collects many one-block messages to guarantee enough pairs which produce a difference with the dBB-condition. Then append a fixed one-block message to the collected messages, query their MACs, and try to find out a dBB-collision. The main idea of the distinguishing attack is summarized as follows:

To maintain the appearance of a dBB-collision under the dBB-condition, i. e., the dBB pseudo-collision happens in the second iteration, the adversary selects a structure S composed of 2^{89} one-block messages P. Then a fixed 447-bit message M is appended to each $P \in S$. Query the MACs with all $P\|M$, and find all collision pairs $(P\|M, P'\|M)$ by birthday attack. A dBB -collision is detected as follows.

- If $(P\|M, P'\|M)$ collides, append another M' to (P, P'), and query two MACs for the new pair $(P\|M', P'\|M')$. If two MACs are the same, we conclude that (P, P') is an internal collision.
- If (P, P') dos not collide, append 2^{47} different M' to (P, P'), query their MACs, and search whether there is a collision. If a collision is found, $(\overline{H}(P), \overline{H}(P'))$ must satisfy the dBB-condition, and $(P\|M, P'\|M)$ is a dBB-collision.
- Otherwise, $(\overline{H}(P), \overline{H}(P'))$ are random values.

Once a dBB-collision is detected, it is concluded that the MAC is a MAC based on MD5, otherwise, based on a random function.

3.2 Description of the New Distinguisher

Our attack is motivated by the above idea, which is to distinguish an instantiated LPMAC from a random function by detecting the target inner near-collision. However, our distinguisher is a totally different one. All the previous attacks detect the collision (or near-collision) generated by a full iteration of the hash function [4,6,8,11,12,14,18], but our attack is trying to distinguish an inner near-collision occurring in the first round, and there is no published techniques to detect such a near-collision till now.

For SHA-1 reduced to 61 steps, we consider a differential path with two message blocks, and assume that $P\|M_0\|M_1$ and $P'\|M_0'||M_1'$ are a message pair, which produces a target differential path. Here, P and P' are one-block messages,

M_0 and M_0' are 448-bit (14 words) truncated messages of the second block, and M_1 and M_1' are the corresponding 64-bit messages left. Denote the output of the first iteration $\overline{H}_K(P)$ as h_P, $\overline{H}_K(P')$ as h'_P, and suppose the intermediate chaining variables $(a_i, \cdots, e_i)$ of the second block as ch_i, and $(a'_i, \cdots, e'_i)$ as ch'_i. It is clear that $h_P + ch_{61}$ and $h'_P + ch'_{61}$ are hash values of the message pair.

We select a target differential path, such that the first iteration can be any differential path, and the second includes a near-collision differential path. To make the truncated differential path of the last 47 steps with high probability, we choose the same disturbance vector as [16], which produces the near-collision differential path for the second iteration (See Table 2). We divide the second differential path into two parts, the first part consists of the previous 14 steps which involves most conditions, and the second part is the last 47 steps with only 34 conditions. We neglect the special differential path in previous 14 steps, and only consider its output difference, which can be regarded as an inner near-collision. We select the specific difference $\triangle ch_{14} = ch_{14} \oplus ch'_{14}$ as

$$(0x00000000, 0x00000000, 0x80000000, 0x20000002, 0x00000040).$$

The choice of $\triangle ch_{14}$ is to cancel the message word differences $\triangle w_{14}$ and $\triangle w_{15}$. The sufficient conditions for the cancelation are as follows:

$$\begin{aligned}
&a_{10,9} = w_{14,7} + 1,\\
&a_{11,4} = w_{15,2} + 1, a_{11,32} = w_{14,30},\\
&a_{13,2} = 1, a_{13,30} = 0, a_{13,32} = 1, a_{13,4} = w_{14,2} + w_{16,2} + 1,\\
&a_{14,4} = w_{14,2} + w_{16,2}, a_{14,32} = 0.
\end{aligned}$$

The core of our attack is to explore some mathematical properties that can be used to distinguish the inner near-collision in the 14th step. For the LPMAC, there are two obstacles to do this:

1. In the first iteration, the output difference $\Delta^- H_P = H_P - H'_P$ is unknown, which conceals the difference of the near-collision $\Delta^- H_P + \Delta^- ch_{61}$. Hence, the birthday attack can not be applied directly to the second iteration like the distinguishing attacks on MACs based on MD5.
2. How to choose messages, and fulfill the birthday attack to detect the inner near-collision?

We explore the following mathematical properties of the differential path to surpass the above two obstacles:

- If the inner near-collision occurs, replace (M_1, M_1') with another $(\overline{M}_1,\ \overline{M_1'})$, then $(P\|M_0\|\overline{M}_1,\ P'\|M_0'\|\overline{M_1'})$ follows the differential path with probability 2^{-34}.
- If two pairs $(P\|M_0\|M_1, P'\|M_0'\|M_1')$ and $(P\|M_0\|N_1, P'\|M_0'\|N_1')$ result in the near-collision differential path, i. e.,

$$\begin{aligned}
\overline{H}_K(P\|M_0\|M_1) - \overline{H}_K(P'\|M_0'\|M_1') &= \overline{H}_K(P\|M_0\|N_1) - \overline{H}_K(P'\|M_0'\|N_1')\\
&= \Delta^- H_P + \Delta^- ch_{61} = \delta, \qquad (1)
\end{aligned}$$

Table 2. A Differential Path for Steps 1∼61 on SHA-1

step i	disturb. vector	XOR difference of the input to step i Δw_{i-1}	Δa_i	Δb_i	Δc_i	Δd_i	Δe_i	conditions
1	80000001	2, 7, 31, 32	-	-	-	-	-	-
2	2	5, 6	-	-	-	-	-	-
3	3	2, 7, 30, 31, 32	-	-	-	-	-	-
4	2	6, 7, 30	-	-	-	-	-	-
5	80000002	1, 7, 30, 31, 32	-	-	-	-	-	-
6	2	5, 7, 30	-	-	-	-	-	-
7	80000003	1, 7, 31, 32	-	-	-	-	-	-
8	0	2, 5, 6, 7, 30, 31, 32	-	-	-	-	-	-
9	80000000	1, 2, 30, 32	-	-	-	-	-	-
10	2	2, 5, 31, 32	9	-	-	-	-	-
11	80000001	1, 7, 30, 31	4, 32	9	-	-	-	-
12	0	2, 5, 6, 31, 32	2	4,32	7	-	-	-
13	0	1, 30		2	2, 30	7	-	-
14	2	2, 31, 32			32	2, 30	7	$a_{13,2} = 1, a_{13,30} = 0, a_{13,32} = 1$ $a_{11,4} = w_{15,2} + 1, a_{11,32} = w_{14,30}, a_{10,9} = w_{14,7} + 1$
15	2	2, 7, 30, 31, 32	2			32	2,30	$a_{15,2} = w_{14,2}, a_{14,32} = 1$
16	0	2, 7, 30, 31		2			32	$a_{14,4} = w_{14,2} + w_{16,2}, a_{13,4} = w_{14,2} + w_{16,2} + 1$
17	0	2, 32			32			$a_{16,32} = 0$
18	0					32		$a_{17,32} = 1$
19	0						32	
20	0	32						
21	2	2	2					$a_{21,2} = w_{20,2}$
22	0	7		2				$a_{20,4} = a_{19,4} + w_{20,2} + w_{23,7} + 1$
23	2		2		32			$a_{23,2} = w_{23,7} + 1$
24	0	7,32		2		32		$a_{22,4} = a_{21,4} + w_{23,7} + w_{25,7}$
25	2	32	2		32		32	$a_{25,2} = w_{25,7} + 1$
26	0	7		2		32		$a_{24,4} = a_{23,4} + w_{25,7} + w_{26,1}$
27	3	1, 32	1		32		32	$a_{27,1} = w_{26,1} + 1$
28	0	6, 7		1		32		$a_{26,3} = a_{25,3} + w_{26,1} + w_{28,1} + 1$
29	0	1, 2, 32			31		32	$a_{28,31} = a_{26,1} + w_{26,1} + w_{29,31}$
30	2	2,31	2			31		$a_{30,2} = w_{29,2}, a_{29,31} = a_{28,1} + w_{26,1} + w_{30,31} + 1$
31	0	7,31, 32		2			31	$a_{29,4} = a_{28,4} + w_{29,2} + w_{31,2} + 1$
32	0	2, 31, 32			32			
33	0	32				32		
34	0	32					32	
35	2	2,32	2					$a_{35,2} = w_{34,2}$
36	0	7		2				$a_{34,4} = a_{33,4} + w_{34,2} + w_{36,2} + 1$
37	0	2			32			
38	0	32				32		
39	0	32					32	
40	0	32						
41	2	2	2					$a_{41,2} = w_{40,2}$
42	0	7		2				$a_{40,4} = a_{39,4} + 1$
43	2		2		32			$a_{43,2} = w_{41,2}, a_{42,32} = a_{40,2} + 1$
44	0	7, 32		2		32		$a_{42,4} = a_{41,4} + 1, a_{43,32} = a_{42,2} + 1$
45	0	2,32			32		32	$a_{44,32} = a_{42,2} + 1$
46	0					32		$a_{45,32} = a_{44,2} + 1$
47	0	32					32	
48	0	32						
...	0							
54	0							
55	4	3	3					$a_{55,3} = w_{54,3}$
56	0	8		3				$a_{54,5} = a_{53,5} + 1$
57	0	3			1			$a_{56,1} = a_{54,3} + 1$
58	8	1, 4	4			1		$a_{57,1} = a_{56,3} + 1, a_{58,4} = w_{57,4}$
59	4	1, 3, 9	3	4			1	$a_{57,6} = a_{56,6} + 1, a_{59,3} = w_{58,3}$
60	0	1, 4, 8		3	2			$a_{59,2} = a_{57,4} + w_{57,4} + w_{60,2} + 1,$ $a_{58,5} = a_{57,5} + w_{58,3} + w_{60,3} + 1$
61	10	2, 3, 5	5		1	2		

then we have (See Fig. 1.)

$$\overline{H}_k(P\|M_0\|M_1)-\overline{H}_K(P\|M_0\|N_1) = \overline{H}_K(P'\|M_0'\|M_1')-\overline{H}_K(P'\|M_0'\|N_1') = \delta'. \tag{2}$$

We utilize the equation (2) to construct a distinguisher for LPMAC with 61-step SHA-1.

- Firstly, collect enough messages $P\|M_0\|M_1$, $P\|M_0\|N_1$, $P'\|M_0'\ \|M_1'$, $P'\|M_0'\ \|N_1'$ and their MACs. Compute two structures, one structure is
$$S_1 = \{(\overline{H}_K(P\|M_0\|M_1) - \overline{H}_K(P\|M_0\|N_1)\},$$
and the other is
$$S_2 = \{\overline{H}_K(P'\|M_0'\|M_1') - \overline{H}_K(P'\|M_0'\|N_1')\}.$$
- Secondly, apply the birthday attack to the structures S_1 and S_2, and search all the collisions such that
$$\overline{H}_K(P\|M_0\|M_1)-\overline{H}_K(P\|M_0\|N_1) = \overline{H}_K(P'\|M_0'\|M_1')-\overline{H}_K(P'\|M_0'\|N_1').$$
- Finally, for each collision, detect whether the corresponding pair ($P\|M_0\|$ M_1, $P'\|M_0'\|M_1'$) satisfies the differential path in Table 2.

More details about the distinguisher are described in the following section. It is noted that, it is hard to fulfill the birthday attack directly to search the solution to equation (1), but easy to get the solution to equation (2) by birthday attack.

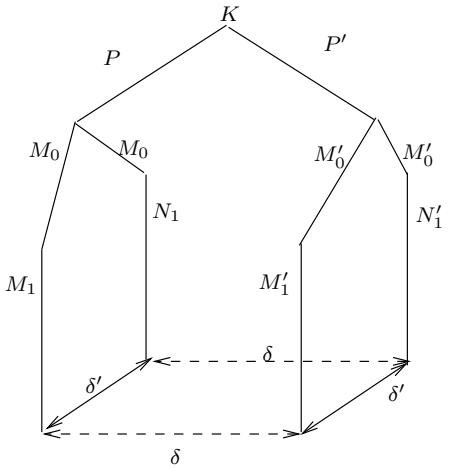

Fig. 1. The Distinguishing Attack on LPMAC

The distinguisher is applicable to LPMACs with other hash functions, such as LPMAC with reduced SHA-2.

4 New Distinguishing Attack on LPMAC Based on 61-step SHA-1

In this section, we describe the new distinguishing attack in detail. We assume that the LPMAC algorithm is either LPMAC with 61-step SHA-1 or LPMAC with a random function.

Before we introduce the new attack, we need to make clear that how many chosen messages are needed to guarantee an inner near-collision. The total sufficient conditions for the near-collision is 169, where 160 conditions are from the difference $\triangle ch_{14}$, and 9 conditions (See Subsection 3.2) are deduced from the cancelation of $\triangle w_{14}$ and $\triangle w_{15}$. So, we need $2^{169/2} = 2^{84.5}$ messages to guarantee such an inner near-collision happen.

Select four messages $M_0\|M_1$, $M_0\|N_1$, $M'_0\|M'_1$ and $M'_0\|N'_1$ such that $\Delta(M_0\|M_1) = (M_0\|M_1) \oplus (M'_0\|M'_1)$ and $\Delta(M_0\|N_1) = (M_0\|N_1) \oplus (M'_0\|N'_1)$ are consistent with the target message difference in Table 2, and $M_0\|M_1$, $M_0\|N_1$ satisfy the sufficient conditions in Table 3.

The distinguishing attack implements the following four steps:

Table 3. Conditions on Messages

$w_{14,31} = w_{14,30} + 1, w_{15,7} = w_{14,2} + 1, w_{15,30} = w_{14,30}, w_{15,31} = w_{15,30} + 1,$
$w_{21,7} = w_{20,2} + 1, w_{27,6} = w_{26,1} + 1, w_{27,7} = w_{26,1}, w_{28,2} = w_{28,1} + 1,$
$w_{30,7} = w_{29,2} + 1, w_{31,31} = w_{26,1} + 1, w_{35,7} = w_{34,2} + 1, w_{41,7} = w_{40,2} + 1,$
$w_{43,7} = w_{40,2} + 1, w_{44,2} = w_{40,2} + 1, w_{55,8} = w_{54,3} + 1, w_{56,3} = w_{54,3} + 1,$
$w_{57,1} = w_{54,3} + 1, w_{58,1} = w_{54,3} + 1, w_{58,9} = w_{57,4} + 1, w_{59,1} = w_{54,3} + 1,$
$w_{59,4} = w_{57,4} + 1, w_{59,8} = w_{58,3} + 1$

1. Randomly choose a structure S, which consists of $2^{84.5}$ different one-block messages.
2. For all $P \in S$, query the MACs with $P\|M_0\|M_1$, $P\|M'_0\|M'_1$, $P\|M_0\|N_1$ and $P\|M'_0\|N'_1$, respectively, and compute the following two structures of differences

$$S_1 = \{LPMAC(P\|M_0\|M_1) - LPMAC(P\|M_0\|N_1) \mid P \in S\},$$
$$S_2 = \{LPMAC(P\|M'_0\|M'_1) - LPMAC(P\|M'_0\|N'_1) \mid P \in S\}.$$

   Search all the collisions between two structures by a birthday attack.
3. For each collision, compute $LPMAC(P\|M_0\|M_1) - LPMAC(P'\|M'_0\|M'_1)$, and denote it as δ. Then for the message pair $(P\|M_0, P'\|M'_0)$, we choose 2^{34} different message pairs $(\overline{M_1}, \overline{M'_1})$ such that $M_0\|\overline{M_1}$ satisfies the message sufficient conditions for the near-collision path in Table 3. Query the MACs for $(P\|M_0\|\overline{M_1}, P'\|M'_0\|\overline{M'_1})$. Check whether the difference $LPMAC(P\|M_0\|\overline{M_1}) - LPMAC(P'\|M'_0\|\overline{M'_1})$ is equivalent to δ.
   - If a pair $(P\|M_0\|\overline{M_1}, P'\|M'_0\|\overline{M'_1})$ that matches the difference δ is searched, we conclude that the LPMAC is based on 61-step SHA-1, and stop the algorithm.

  - Else, go to step 4.
4. Repeat steps 1-3. If the number of structures exceeds 2^{68}, we conclude that the LPMAC is constructed from a random function.

Complexity:

Summing up the above attack, we choose $4 \cdot 2^{68} \cdot (2^{84.5} + 2^{34}) \approx 2^{154.5}$ messages in total. Since we can use the birthday attack to search collisions in step 2, a table with size of $2^{84.5}$ needs to be built. We need about $2^{68} \cdot 2^{84.5} = 2^{152.5}$ table lookups and $2^{154.5}$ queries.

Success rate:

From the above process, the success rate of our attack can be divided into two parts :

- If the LPMAC is constructed from 61-step SHA-1, once a second collision in step 3 is detected, the attack succeeds. The probability is computed as follows:
  - There are 169 conditions to guarantee the inner near-collision in the step 14, and 34 conditions to follow the differential path in the last steps 15-61. According to the birthday paradox and Taylor series expansion, for $2^{68} \cdot 2^{169} = 2^{237}$ pairs among the structures S_1 and S_2, there exists an inner near-collision with probability

$$1 - (1 - \frac{1}{2^{237}})^{2^{237}} \approx 1 - e^{-1} \approx 0.63.$$

  - If the first collision is captured in step 2, the second collision in step 3 is searched with probability

$$1 - (1 - \frac{1}{2^{34}})^{2^{34}} \approx 1 - e^{-1} \approx 0.63.$$

  Hence, when the LPMAC is based on 61-step SHA-1, the distinguishing attack successes with probability

$$0.63 \cdot 0.63 \approx 0.40$$

  .
- If the LPMAC is from a random function, the attack succeeds when the second collision doesn't exist. For the 2^{68} structures, there are $2^{237}/2^{160} = 2^{77}$ expected collisions in total, so the success probability is about:

$$((1 - \frac{1}{2^{160}})^{2^{34}})^{2^{77}} \approx 1.$$

  Therefore, the success rate of the whole attack is about

$$\frac{1}{2} \times 0.40 + \frac{1}{2} \times 1 = 0.70.$$

Note that the success probability can be increased by repeating this attack several times, doubling the size of the structure S or the number of different pairs $(\overline{M_1}, \overline{M_1'})$.

Table 4 illustrates the comparison of our distinguishing attacks on LPMAC-SHA-1 with other attacks on HMAC-SHA-1.

Table 4. Comparison Between the Distinguishing Attacks on MACs with SHA-1

	MAC	steps	data
Kim *et al.* [8]	HMAC	43	$2^{154.9}$
Rechberger *et al.* [12]	HMAC	50	$2^{153.5}$
This paper	LPMAC	43	$2^{124.5}$
		50	$2^{136.5}$
		61	$2^{154.5}$

5 Conclusions

A new distinguisher is introduced to recognize the secret-prefix MAC which prepends the message length before hashing. The new distinguisher utilizes a near-collision differential path instead of a collision path, and detects an inner near-collision in the first round. The core of the attack is to capture the mathematical characters of a near-collision differential path which can be utilized to fulfill a birthday attack. Our attack is applicable to some other LPMACs such as LPMAC with reduced SHA-2.

Acknowledgements. We would like to thank Christian Rechberger and three reviewers for their very helpful comments on the paper.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
2. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
3. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and Reduced SHA-1. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 36–57. Springer, Heidelberg (2005)
4. Contini, S., Yin, Y.L.: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
5. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)

6. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
7. Galvin, J.M., McCloghrie, K., Davin, J.R.: Secure Management of SNMP Networks. Integrated Network Management, II, pp. 703–714 (1991)
8. Kim, J., Biryukov, A., Preneel, B., Hong, S.: On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0, and SHA-1. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
9. NIST: Secure Hash Standard. Federal Information Processing Standard, FIPS-180-1 (April 1995)
10. Preneel, B., and van Oorschot, P.: MDx-MAC and Building Fast MACs from Hash Functions. In: Coppersmith,D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14, Springer, Heidelberg (1995)
11. Rechberger, C., Rijmen, V.: On Authentication with HMAC and Non-random Properties. In: Dietrich, S., Dhamija, R. (eds.) Financial Cryptography 2007. LNCS, vol. 4886, pp. 119–133. Springer, Heidelberg (2007)
12. Rechberger, C., Rijmen, V.: New Results on NMAC/HMAC when Instantiated with Popular Hash Functions. Journal of Universal Computer Science, 14(3), pp. 347–376 (2008)
13. Tsudik, G.: Message Authentication with One-Way Hash Functions. ACM Comput. Commun. Rev., 22 (5), pp. 29–38 (1992)
14. Wang, L., Ohta, K., Kunihiro, N.: New Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 237–253. Springer, Heidelberg (2008)
15. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
16. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
17. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
18. Wang, X., Yu, H., Wang, W., Zhang, H., Zhan, T.: Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol 5479, pp. 121-133. Springer, Heidelberg (2009)
19. Wang, X., Yu, H., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)
20. Yu, H., Wang, G., Zhang, G., Wang, X.: The Second-Preimage Attack on MD4. In: Desmedt, Y., Wang, H., Mu, Y., Li, Y. (Eds.) CANS 2005, LNCS vol. 3810, pp. 1–12. Springer, Heidelberg (2005)