A shortened version of this paper will appear under the same title
in the proceedings of *SCN 2006* (Moti Yung ed.), *LNCS*, Springer-Verlag.

On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1*

Jongsung Kim[1]**, Alex Biryukov[2], Bart Preneel[1], and Seokhie Hong[3]

[1] ESAT/SCD-COSIC, Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{Kim.Jongsung,Bart.Preneel}@esat.kuleuven.be
[2] FDEF, Campus Limpertsberg, University of Luxembourg,
162 A, Avenue de la Faiencerie L-1511 Luxembourg
alex.biryukov@uni.lu
[3] Center for Information Security Technologies(CIST),
Korea University, Seoul, Korea
hsh@cist.korea.ac.kr

Abstract. HMAC is a widely used message authentication code and a pseudorandom function generator based on cryptographic hash functions such as MD5 and SHA-1. It has been standardized by ANSI, IETF, ISO and NIST. HMAC is proved to be secure as long as the compression function of the underlying hash function is a pseudorandom function. In this paper we devise two new distinguishers of the structure of HMAC, called *differential* and *rectangle distinguishers*, and use them to discuss the security of HMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. We show how to distinguish HMAC with reduced or full versions of these cryptographic hash functions from a random function or from HMAC with a random function. We also show how to use our differential distinguisher to devise a forgery attack on HMAC. Our distinguishing and forgery attacks can also be mounted on NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. Furthermore, we show that our differential and rectangle distinguishers can lead to second-preimage attacks on HMAC and NMAC.

1 Introduction

Message Authentication Code (MAC) algorithms are widely used in Internet security protocols (SSL/TLS, SSH, IPsec) and in the financial sector for debit and credit transactions. MAC algorithms are keyed hash functions that allow to verify whether a transmitted message has been altered. In order to use a MAC algorithm in computer networks, a secret key should be first distributed to the authorized entities, Alice and Bob. When Alice sends a message to Bob, she computes the MAC value of the message with the shared secret key and appends it to the message. Once Bob receives the message and its MAC value, he recomputes the MAC value of the obtained message with the key and verifies the authenticity of the message by checking if the recomputed MAC value is the same as the received MAC value. The security

* This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the European Commission through the IST Programme under Contract IST2002507932 ECRYPT and in part by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

** The first author was financed by a Ph.D grant of the Katholieke Universiteit Leuven and by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD) (KRF-2005-213-D00077).

of a MAC algorithm depends on the difficulty for an unauthorized entity to produce a forgery, that is, a new message with a valid MAC. Typically, the forger is allowed to query the MAC generation oracle with adaptively chosen queries (see for example [3, 17]).

In the literature there have been mainly two types of MAC algorithms: block cipher based MAC algorithms (e.g., CBC-MAC [11], TMAC [15], RMAC [13] and OMAC [12]) and hash function based MAC algorithms (NMAC [2], HMAC [2] and MDx-MAC [18]). Both types of MAC algorithms usually inherit the security and efficiency of its underlying primitives. Other MAC algorithms are based on the design principle of a stream cipher (e.g., SOBER [21]) and a universal hash function (e.g., UMAC [9] and Poly1305–AES MAC [5]). Furthermore, several authenticated encryption schemes have been proposed that offer both confidentiality and authenticity of a message (e.g., CWC [14], EAX [4] and GCM [16]).

HMAC, which was designed by Bellare, Canetti and Krawczyk, is a standardized hash-based MAC algorithm that is widely used as a MAC algorithm and as a pseudorandom function generator [2]. HMAC takes a message of an arbitrary bit-length and hashes it with one secret key. For the same length of the message it calls the compression function of the underlying hash function additionally three more times than the iterated hash construction, i.e., the MD construction. For long messages, its efficiency is thus almost the same as the MD construction. Furthermore, cryptographic hash functions such as MD5 and SHA-1 can be used in HMAC, which are more efficient in software than block ciphers, and thus HMAC is typically faster than block cipher based MACs. HMAC is proved to be a pseudorandom function under the assumption that the compression function of the underlying hash function is a pseudorandom function [1] (note that the security proof of pseudorandomness provides the MAC security [3]). However, this does not guarantee the security of HMAC if it is instantiated with a specific cryptographic hash function such as MD5 or SHA-1. The recent attacks of Wang et al. [22–26] and Biham et al. [7, 8] have undermined the confidence in the most popular collision resistant hash functions such as MD5 and SHA-1. However, it is widely assumed that these attacks have no impact on the security of MAC algorithms based on these hash functions such as HMAC since they use a keyed initial value.

This paper is the first work which presents a detailed analysis of distinguishing and forgery attacks on HMAC based on MD5, SHA-1 and other MDx-type hash functions. Our results allow to quantify to which extent the vulnerabilities of these hash functions carry over to the HMAC construction. This is achieved by the introduction of two novel distinguishers of the general structure of HMAC. We use a message pair which induces a collision in its corresponding MAC pair for designing *a differential distinguisher* of HMAC and also use a message quartet which induces two collisions in its corresponding MAC quartet for designing *a rectangle distinguisher* of HMAC. With these two distinguishers we discuss the security of HMAC based on HAVAL [27], MD4 [19], MD5 [20], SHA-0 [28] and SHA-1 [29].

First, we construct new differentials of the full 3-pass HAVAL and reduced MD5 to form rectangle distinguishers of HMAC, and we use them to distinguish HMAC with the full 3-pass HAVAL and reduced MD5 from HMAC with a random function. Second, we investigate how effectively the differentials of MD4, SHA-0 and SHA-1 found by Wang et al. [22–26] and Biham et al. [7, 8] are applied to our differential and rectangle distinguishers in HMAC. After converting their differentials into our differential and rectangle distinguishers, we devise distinguishing and forgery attacks on HMAC based on reduced or full versions of MD4, SHA-0 and SHA-1. In particular, we show how to distinguish HMAC with the full SHA-0 and MD4 from HMAC with a random function and present a forgery attack on HMAC with the full MD4. See for details of the results Table 3 in Sect. 6 (the function h_2 and the probabilities $\hat{p}$ and q in Table 3 will be defined in the following sections). Our distinguishing and forgery attacks can be mounted on NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1 with the same complexity. Furthermore, we show that our differential and rectangle distinguishers can lead to second-preimage attacks on HMAC and NMAC.

The paper is organized as follows: in Sect. 2, we give a brief description of HMAC and in Sect. 3, we describe some general attacks on HMAC. Section 4 devises two distinguishers of

the general structure of HMAC and Sect. 5 presents differentials on the compression functions of HAVAL, MD4, MD5, SHA-0 and SHA-1, which can be used in our distinguishers of HMAC. In Sect. 6, we present distinguishing and forgery attacks on HMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. We also discuss distinguishing and forgery attacks on NMAC in Sect. 7 and we present some implications of our differential and rectangle distinguishers in Sect. 8. Section 9 concludes the paper.

2 Description of HMAC

HMAC [2] applies in both its inner and outer parts the iterated MD construction of a hash function H given a compression function h, $H(IV, M) = h(\cdots h(h(IV, M^1), M^2) \cdots, M^n)$, where IV is a l-bit fixed initial value and M is an arbitrary-length message which is padded to a multiple of b-bit and divided into n b-bit blocks $M^1||M^2||\cdots||M^n$ (note that the outputs of functions h and H are l-bit strings).

$$\begin{aligned}\mathbf{HMAC}(K, M) &= H(IV, (K \oplus opad)||H(IV, (K \oplus ipad)||M)) \\ &= h(h(IV, (K \oplus opad)), H(h(IV, (K \oplus ipad)), M))\,, \end{aligned} \tag{1}$$

where K is the secret key, $opad$, $ipad$ are constants and $|K \oplus opad| = |K \oplus ipad| = b$. If HMAC takes a one-block message M, it can be expressed as

$$\mathbf{HMAC}(K, M) = h(h(IV, (K \oplus opad)), h(h(IV, (K \oplus ipad)), M))\,. \tag{2}$$

In order to facilitate the description of our analysis of HMAC we denote the four compression functions h in (2) by h_1, h_2, h_3 and h_4, and the four functions in (1) by h_1, H_2, h_3 and h_4. See Fig. 1 for a schematic description of HMAC with this notation. Note that the outputs of H_2 and h_2 are padded to a b-bit string to be inserted into h_4.

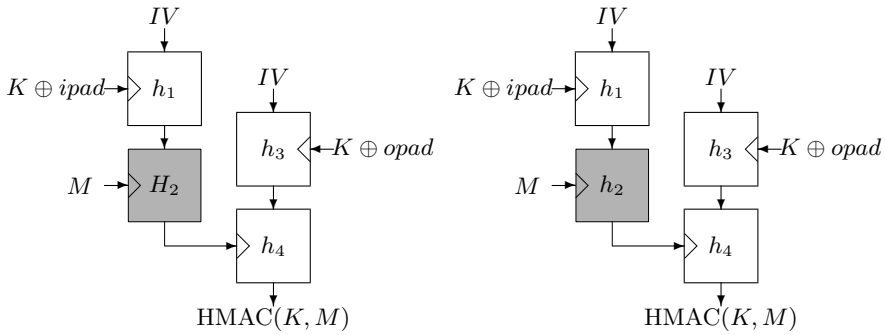

Fig. 1. A schematic description of HMAC

In practice the function h can be replaced by the compression function of cryptographic hash functions such as HAVAL [27], MD4 [19], MD5 [20], SHA-0 [28], SHA-1 [29] and so on. Each compression function of these cryptographic hash functions has the following form:

$$h(I, M) = E(I, M) + I\,, \tag{3}$$

where I is an l-bit initial value, M is a b-bit message block and E is an iterative step function. The function E typically consists of 3, 4 or 5 passes and each form has 16, 20 or 32 rounds.

In HAVAL, MD4, MD5, SHA-0 and SHA-1, the l-bit initial value is loaded into $l/32$ 32-bit registers, $(A, B, C, D, \cdots)$. The b-bit message block M is also loaded into $b/32$ 32-bit words, $(m^0, m^1, m^2, m^3, \cdots)$ and a message expansion algorithm is used to expand the $b/32$ 32-bit words into 32-bit message words (as many as the number of rounds). The l-bit initial value loaded into $l/32$ 32-bit registers is then updated through a number of rounds by using the expanded 32-bit message words. In each pass a fixed Boolean function and fixed 32-bit constants are used. Table 1 shows the parameters of HAVAL, MD4, MD5, SHA-0 and SHA-1 (see [27, 19, 20, 28, 29] for their detailed descriptions).

Table 1. Parameters of HAVAL, MD4, MD5, SHA-0 and SHA-1

Hash Functions	Bit-Length of Message Block (b)	Bit-Length of Initial Value (l)	# of Passes	# of Rounds in a Pass	Total # of Rounds
HAVAL	1024	256	3,4 or 5	32	96, 128 or 160
MD4	512	128	3	16	48
MD5	512	128	4	16	64
SHA-0	512	160	4	20	80
SHA-1	512	160	4	20	80

3 Some General Attacks on HMAC

Using the birthday paradox we can induce a general distinguishing attack on HMAC as follows [18]:

1. Collect $2^{l/2}$ randomly chosen messages with a b-bit length, denoted M_i, and ask for their MAC values, denoted C_i.
2. Find message pairs M_j and M_k such that $C_j = C_k$.
3. For each of (M_j, M_k) pairs such that $C_j = C_k$, ask for a MAC pair of $M_j||P$ and $M_k||P$, where P is some non-empty string. If there is at least one MAC pair that collides in this step, output the MAC algorithm = HMAC.

This attack requires about $2^{l/2}$ messages and works with a probability of 0.63 by the birthday paradox when the MAC algorithm is HMAC (this is due to the fact that if there exists at least one message pair (M_j, M_k) such that their outputs of h_2 or H_2 are same, this attack always works). This attack can also easily be converted into a general forgery attack on HMAC. Once we get a MAC pair that collides in Step 3, we again ask for the corresponding MAC of $M_j||P||P'$, denoted C, where P' is some non-empty string. We can then construct a forgery, i.e., a new message $M_k||P||P'$ with a valid MAC, i.e., C with the same success rate.

These general attacks make distinguishing and forgery attacks on HMAC which require more than $2^{l/2}$ message queries have not much advantage. We thus consider attacks of distinguishing HMAC from a random function, and forgery attacks on HMAC which work with a data complexity of less than $2^{l/2}$ messages. In addition to these two kinds of attacks, we also consider attacks of distinguishing instantiated HMAC (by existing hash functions) from HMAC with a random function. In these attacks it does not matter whether or not they require more than $2^{l/2}$ message queries, since there does not exist a general attack based on the birthday paradox which can distinguish HMAC with existing hash functions from HMAC with a random function. For the clarification we denote the first and second distinguishing attacks by *distinguishing-R* and *distinguishing-H attacks*, respectively. The distinguishing-R attack is useful when the cryptanalyst wants to check whether output strings are produced from HMAC (in this case, the cryptanalyst does not know whether the output producing algorithm is HMAC), while the distinguishing-H attack is useful when the cryptanalyst wants to check which cryptographic hash function is embedded in HMAC (in this case, the cryptanalyst somehow already knew that the output producing algorithm is HMAC, for instance, by the distinguishing-R attack, but does not know the underlying hash function in HMAC).

4 Distinguishers of HMAC

In this section we present two distinguishers of the general structure of HMAC, which can lead to distinguishing or forgery attacks if HMAC is instantiated with some cryptographic hash function with a low difference propagation. These two distinguishers, called *differential* and *rectangle distinguishers*, are both built based on internal collisions. We focus on HMAC with one-block messages, which is the main target in our attacks.

4.1 Differential Distinguisher of HMAC

By using MAC collisions we construct a differential distinguisher of HMAC. It works as follows:

- Choose a message M_i at random and compute another message $M_i' = M_i \oplus \alpha$, where M_i has the same length as α ($\neq 0$).
- With a chosen message attack scenario, obtain the MAC values $C_i = \text{HMAC}(K, M_i)$ and $C_i' = \text{HMAC}(K, M_i')$.
- Check if $C_i \oplus C_i' = 0$.

Assuming that the values $h_1(IV, K \oplus ipad)$ are uniformly distributed for a given key K, the last test holds with a probability[1] of approximately q, where $q = Pr_{X,I}[h_2(I, X) \oplus h_2(I, X \oplus \alpha) = 0]$. On the other hand, for a random function or HMAC with a random function[2], the last test holds with a probability of approximately 2^{-l}. Hence, we have the following differential distinguisher of HMAC.

Proposition 1. *[A Differential Distinguisher of HMAC] Assume that the output values of the function h_1 are distributed uniformly at random. Then HMAC can be distinguished from a random function and from HMAC with a random function if $q > 2^{-l}$, where $q = Pr_{X,I}[h_2(I, X) \oplus h_2(I, X \oplus \alpha) = 0]$.*

In order for this differential distinguisher to be used in distinguishing-R and forgery attacks, the probability q should be larger than $2^{-l/2}$, which makes possible for those attacks to work with less than $2^{l/2}$ message queries (details are described in Sect. 6).

4.2 Rectangle Distinguisher of HMAC

The rectangle distinguisher of HMAC can be built by the rectangle attack which is widely used in analyzing block ciphers [6]. In block ciphers the rectangle attack can be mounted based on their bijectivity. However, in MACs it can exploit the non-bijectivity, i.e., two different messages may correspond to a same MAC value or a same intermediate value (an internal collision). We use this non-bijective property to devise our rectangle distinguisher of HMAC. Our rectangle distinguisher of HMAC works as follows (refer to Fig. 2):

- Choose two messages M_i and M_j at random and compute two other messages $M_i' = M_i \oplus \alpha$ and $M_j' = M_j \oplus \alpha$, where M_i and M_j both have the same length as α ($\neq 0$).
- With a chosen message attack scenario, obtain the MAC values $C_i = \text{HMAC}(K, M_i)$, $C_i' = \text{HMAC}(K, M_i')$, $C_j = \text{HMAC}(K, M_j)$ and $C_j' = \text{HMAC}(K, M_j')$.
- Check if $C_i \oplus C_j = C_i' \oplus C_j' = 0$ or $C_i \oplus C_j' = C_i' \oplus C_j = 0$.

[1] In fact, the last test holds with a probability of approximately $q + (1 - q) \cdot 2^{-l}$. Because even if the M_i and M_i' do not cause a collision after the function h_2, their MAC values can still have a same value. However, in the computation of a probability for our differential distinguisher we do not consider this case.

[2] From [1] we know that HMAC with a random function behaves like a random function.

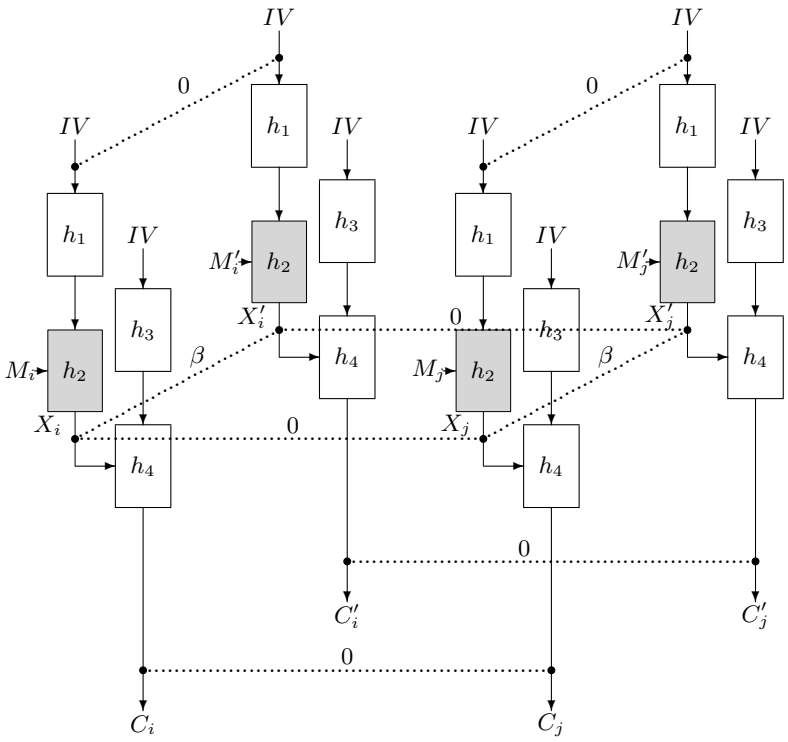

Fig. 2. A Rectangle Distinguisher of HMAC ($M_i \oplus M'_i = M_j \oplus M'_j = \alpha$)

We denote by $X_i,\ X'_i,\ X_j$ and X'_j the outputs of $h_2 \circ h_1$ for the messages $M_i,\ M'_i,\ M_j$ and M'_j, respectively (see Fig. 2). Note that in Fig. 2 $K \oplus ipad$ and $K \oplus opad$ are inserted into the message parts of the functions h_1 and h_3, respectively. In order to compute the probability to satisfy the last test we should consider the following probabilities: $p = Pr_{X,I}[h_2(I, X) \oplus h_2(I, X \oplus \alpha) = \beta]$ and $\hat{p} = \sqrt{\sum_\beta (p^2)}$.

Assuming that the values $h_1(IV, K \oplus ipad)$ are uniformly distributed for a given key K, we get $X_i \oplus X'_i = X_j \oplus X'_j = \beta$ with probability p^2. Since the function h_2 is not a permutation (here, the domain of h_2 is the message space and its co-domain is the space of hash values), we expect $X_i \oplus X_j = 0$ with probability 2^{-l} under the assumption that the output values of h_2 are distributed uniformly at random. Once we get $X_i \oplus X'_i = X_j \oplus X'_j = \beta$ and $X_i \oplus X_j = 0$, we have the following equation:

$$X'_i \oplus X'_j = (X_i \oplus \beta) \oplus (X_j \oplus \beta) = X_i \oplus X_j = 0$$

These equations allow us to get $C_i \oplus C_j = C'_i \oplus C'_j = 0$ and thus the probability[3] of satisfying $C_i \oplus C_j = C'_i \oplus C'_j = 0$ is approximately

$$\sum_\beta p^2 \cdot 2^{-l} = \hat{p}^2 \cdot 2^{-l}\,.$$

Similarly, we get $X_i \oplus X'_j = 0$ with a probability of 2^{-l} and thus $C_i \oplus C'_j = C'_i \oplus C_j = 0$ holds with the same probability $\hat{p}^2 \cdot 2^{-l}$.

[3] Note that the probability of satisfying $C_i \oplus C_j = C'_i \oplus C'_j = 0$ is slightly larger than $\hat{p}^2 \cdot 2^{-l}$. Because even if the X_i and X_j (or the X'_i and X'_j) are not the same, still there is a chance to have $C_i \oplus C_j = C'_i \oplus C'_j = 0$. However, we believe that a simplified analysis is sufficient for the computation of the probability for our rectangle distinguisher.

On the other hand, for a random function or HMAC with a random function, $C_i \oplus C_j = C'_i \oplus C'_j = 0$ and $C_i \oplus C'_j = C'_i \oplus C_j = 0$ hold with a probability of approximately 2^{-2l}, respectively, since each requires a $2l$-bit restriction to be satisfied. Hence, we have the following rectangle distinguisher of HMAC.

Proposition 2. *[A Rectangle Distinguisher of HMAC] Assume that the output values of the functions h_1 and H_2 are distributed uniformly at random. Then HMAC can be distinguished from a random function and from HMAC with a random function if $\hat{p}^2 \cdot 2^{-l} > 2^{-2l}$, i.e., $\hat{p} > 2^{-l/2}$, where $\hat{p} = \sqrt{\sum_\beta (p^2)}$ and $p = Pr_{X,I}[h_2(I, X) \oplus h_2(I, X \oplus \alpha) = \beta]$.*

Our rectangle distinguisher cannot be used in distinguishing-R and forgery attacks, since its required data complexity is always larger than $2^{l/2}$ messages (details are described in Sect. 6). This is due to the fact that the rectangle probability is always less than or equal to 2^{-l}.

Unlike the differential distinguisher of HMAC, the rectangle distinguisher uses a number of differentials without any restriction for output differences, while its requirement to work is more expensive than that of the differential distinguisher, i.e., it uses probability $2^{-l/2}$ instead of 2^{-l} for its comparison. If it is easy to get some nonzero output difference from the compression function of the underlying hash function, but it is difficult to get a zero output difference, i.e., a collision, then this rectangle distinguisher would be useful.

The success of our two distinguishers for HMAC depends significantly on the strength of h_2, which means the distinguishers do not depend strongly on the properties of h_1, h_3 and h_4. Even if h_1, h_3 and h_4 employs cryptographically strong compression functions (even iterated hash functions), our distinguishers can still work if h_2 has a low difference propagation.

5 Differentials on HAVAL, MD4, MD5, SHA-0 and SHA-1

First, we check how many rounds of the compression functions of HAVAL, MD4, MD5, SHA-0 and SHA-1 can be used for h_2 in our rectangle distinguisher, i.e., we investigate for how many rounds of each compression function $\hat{p} > 2^{-l/2}$ holds. Second, we discuss how to extend one-block messages (corresponding to h_2) into multi-block messages (corresponding to H_2) to apply them to our rectangle distinguisher. Third, we deal with differentials with probabilities q such that $q > 2^{-l}$ or $q > 2^{-l/2}$.

5.1 One-Block Differentials for Rectangle Distinguishers

In order to compute the number of rounds for each compression function such that $\hat{p} > 2^{-l/2}$, we investigate a differential with probability p from which the probability $\hat{p}$ can be estimated. We first consider the compression function of 3-pass HAVAL.

In the compression function of HAVAL we insert a one-bit difference to two message words to produce a collision after the first pass with a high probability. This enables us to get probability-one differentials through many rounds in the first and second passes. In more detail, if we denote by r_1, r_2, r_3, r_4, r_5 and r_6 the round numbers involved in two such message words in the three passes where $r_1 < r_2 < \cdots < r_6$, we can construct a 96-round differential with the following probability: for the rounds $0 \sim r_1$ probability 1, for each of the rounds $(r_1 + 1) \sim r_2$ probability 2^{-1}, for the rounds $(r_2 + 1) \sim (r_3 - 1)$ probability 1, for each of the rounds $r_3 \sim r_4$ probability 2^{-1}, for each of the rounds $(r_4 + 1) \sim (r_5 - 1)$ probability 2^{-2}, for each of the rounds $r_5 \sim r_6$ probability 2^{-3} and for each of the rounds $(r_6+1) \sim 95$ probability 2^{-4} (this can be done by the computation of differential probabilities derived from the differential distributions of Boolean functions and the use of both XOR and modular additions). These probabilities may be slightly different according to in which message word between the two a difference $0x80000000$ is given. But the total probability is the same: $2^{-(r_2-r_1+r_4-r_3+1+2(r_5-r_4-1)+3(r_6-r_5+1)+4(95-r_6))}$.

As a result of an exhaustive search[4], inserting a one-bit difference to the third and eleventh message words provides the best probability $p = 2^{-102}$. See Table 2 for more details. In Table 2 e_i represents a 32-bit word that has $0's$ in all bit positions except for bit i and $e_{i_1,\cdots,i_k}$ represents $e_{i_1} \oplus \cdots \oplus e_{i_k}$ (in our notation the leftmost bit is referred to as the 31-th bit, i.e., the most significant bit). Note that we use the XOR difference as the measure of difference and in the computation of the probability p in Table 2 the modular additions of the unknown initial value and the last output value are considered. In our analysis we take into account the probability that the last output difference is preserved through the final modular additions.

In order to calculate $\hat{p}$ we should sum the square of the probability of all differentials with message difference α. However, it is computationally infeasible and thus we have carried out experiments on the last three rounds (rounds $93 \sim 95$) to estimate a lower bound for $\hat{p}$ (our simulation is based on the assumption that chosen message pairs follow the first 93-round differential in Table 2). For this work, we have randomly chosen a number of IVs with 2^{28} message pairs $M_i,\ M_i^*$ and 2^{28} input pairs of round 93 $I_i,\ I_i^*$ each and computed $M_i^{'} = M_i \oplus \alpha$, $M_i^{*'} = M_i^* \oplus \alpha$ and $I_i^{'} = I_i \oplus \delta$ and $I_i^{*'} = I_i^* \oplus \delta$, where α is the message difference and δ is the input difference of round 93 in Table 2. We have then encrypted through rounds $93 \sim 95$ $I_i,\ I_i^{'},\ I_i^*$ and $I_i^{*'}$ with $M_i,\ M_i^{'},\ M_i^*$ and $M_i^{*'}$ to obtain outputs $O_i,\ O_i^{'},\ O_i^*$ and $O_i^{*'}$. Finally, we have checked if $(O_i + IV) \oplus (O_i^{'} + IV) = (O_i^* + IV) \oplus (O_i^{*'} + IV)$. In our experiments we have observed that the number of such quartets was ranging $320 \sim 2130$ for each IV. This simulation result suggests that the square of the probability $\hat{p}$ for rounds $93 \sim 95$ is approximately $2^{-18.2}$ and thus we can estimate the probability $\hat{p} \approx 2^{-9.1} \cdot 2^{-90} = 2^{-99.1}$ since the differential probability for rounds $0 \sim 92$ in Table 2 is 2^{-90}. Furthermore, we can extend this differential up to 101 rounds such that $\hat{p} > 2^{-128}$. See Table 2 for this extension. We have also performed a series of simulations on the last two rounds and from the simulation result we can estimate $\hat{p} \approx 2^{-124.4}$ for rounds $0 \sim 101$.

Similarly, we have investigated differentials on the compression function of MD5 with high probabilities by inserting a one-bit difference in two or three message words to produce a collision after the first pass. As a result, we can construct a 33-round differential on MD5 with probability $p = 2^{-69}$, which can be used to construct differentials with probability $\hat{p}$. See Table 4 in Appendix A for details of our reduced MD5 differential. Our investigations on HAVAL and MD5 have started from the assumption that low-weight differentials work out best when we can not use neutral bits and message modifications. However, still there is a possibility that HAVAL and MD5 have stronger differentials which can be derived by other methods.

For MD4, SHA-0 and SHA-1, we have used the previous differentials in our distinguishers, i.e., a 48-round differential on MD4 with probability 2^{-56} in [26], a 65-round differential on SHA-0 with probability 2^{-78} in [7, 8] and a 43-round differential on SHA-1 with probability 2^{-80} in [8]. The 43-round differential on SHA-1 is an extended one for the 34-round differential described in [8], and the computations of differential probabilities on SHA-0 and SHA-1 are recomputed[5]. See Appendix A for the recomputed differentials of SHA-0 and SHA-1. We have also carried out the same experiments on the last few rounds to estimate each $\hat{p}$ and from our simulations we can estimate $\hat{p} \approx 2^{-56},\ 2^{-60.6},\ 2^{-78}$ and $2^{-73.4}$ for 48-round MD4, 33-round MD5, 65-round SHA-0 and 43-round SHA-1, respectively.

[4] The exhaustive search has experimentally been done by considering all possible $r_1,\ r_2,\ r_3,\ r_4,\ r_5$ and r_6 which can produce a collision after the first pass.

[5] The main difference of the computations of differential probabilities between [7, 8] and this paper is the use of neutral bits. In the SHA-0 and SHA-1 initial values are known, which enables us to use neutral bits on message pairs to improve differential probabilities. However, in our analysis of HMAC initial values are determined by a secret key K, which implies they are unknown.

Table 2. A Differential of HAVAL

Round (i)	ΔA^i	ΔB^i	ΔC^i	ΔD^i	ΔE^i	ΔF^i	ΔG^i	ΔH^i	Δm^i	Prob.
0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	$e_{31}(=\Delta m^2)$	1
3	0	0	0	0	0	0	0	e_{31}	0	2^{-1}
4	0	0	0	0	0	0	e_{31}	0	0	2^{-1}
5	0	0	0	0	0	e_{31}	0	0	0	2^{-1}
6	0	0	0	0	e_{31}	0	0	0	0	2^{-1}
7	0	0	0	e_{31}	0	0	0	0	0	2^{-1}
8	0	0	e_{31}	0	0	0	0	0	0	2^{-1}
9	0	e_{31}	0	0	0	0	0	0	0	2^{-1}
10	e_{31}	0	0	0	0	0	0	0	$e_{20}(=\Delta m^{10})$	2^{-1}
11	0	0	0	0	0	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	0	0	0	0	0	0	0	0	0	1
45	0	0	0	0	0	0	0	0	$e_{20}(=\Delta m^{10})$	2^{-1}
46	0	0	0	0	0	0	0	e_{20}	0	2^{-1}
47	0	0	0	0	0	0	e_{20}	0	0	2^{-1}
48	0	0	0	0	0	e_{20}	0	0	0	2^{-1}
49	0	0	0	0	e_{20}	0	0	0	0	2^{-1}
50	0	0	0	e_{20}	0	0	0	0	0	2^{-1}
51	0	0	e_{20}	0	0	0	0	0	0	2^{-1}
52	0	e_{20}	0	0	0	0	0	0	0	2^{-1}
53	e_{20}	0	0	0	0	0	0	0	0	2^{-1}
54	0	0	0	0	0	0	0	e_9	0	2^{-1}
55	0	0	0	0	0	0	e_9	0	0	2^{-1}
56	0	0	0	0	0	e_9	0	0	0	2^{-1}
57	0	0	0	0	e_9	0	0	0	0	2^{-1}
58	0	0	0	e_9	0	0	0	0	0	2^{-1}
59	0	0	e_9	0	0	0	0	0	0	2^{-1}
60	0	e_9	0	0	0	0	0	0	$e_{31}(=\Delta m^2)$	2^{-1}
61	e_9	0	0	0	0	0	0	e_{31}	0	2^{-2}
62	0	0	0	0	0	0	e_{31}	e_{30}	0	2^{-2}
63	0	0	0	0	0	e_{31}	e_{30}	0	0	2^{-2}
64	0	0	0	0	e_{31}	e_{30}	0	0	0	2^{-2}
65	0	0	0	e_{31}	e_{30}	0	0	0	0	2^{-2}
66	0	0	e_{31}	e_{30}	0	0	0	0	0	2^{-2}
67	0	e_{31}	e_{30}	0	0	0	0	0	0	2^{-2}
68	e_{31}	e_{30}	0	0	0	0	0	0	0	2^{-2}
69	e_{30}	0	0	0	0	0	0	e_{20}	0	2^{-2}
70	0	0	0	0	0	0	e_{20}	e_{19}	0	2^{-2}
71	0	0	0	0	0	e_{20}	e_{19}	0	0	2^{-2}
72	0	0	0	0	e_{20}	e_{19}	0	0	0	2^{-2}
73	0	0	0	e_{20}	e_{19}	0	0	0	0	2^{-2}
74	0	0	e_{20}	e_{19}	0	0	0	0	0	2^{-2}
75	0	e_{20}	e_{19}	0	0	0	0	0	0	2^{-2}
76	e_{20}	e_{19}	0	0	0	0	0	0	0	2^{-2}
77	e_{19}	0	0	0	0	0	0	e_9	0	2^{-2}
78	0	0	0	0	0	0	e_9	e_8	0	2^{-2}
79	0	0	0	0	0	e_9	e_8	0	0	2^{-2}
80	0	0	0	0	e_9	e_8	0	0	0	2^{-2}
81	0	0	0	e_9	e_8	0	0	0	0	2^{-2}
82	0	0	e_9	e_8	0	0	0	0	0	2^{-2}
83	0	e_9	e_8	0	0	0	0	0	0	2^{-2}
84	e_9	e_8	0	0	0	0	0	0	0	2^{-2}
85	e_8	0	0	0	0	0	0	e_{30}	0	2^{-2}
86	0	0	0	0	0	0	e_{30}	e_{29}	0	2^{-2}
87	0	0	0	0	0	e_{30}	e_{29}	0	0	2^{-2}
88	0	0	0	0	e_{30}	e_{29}	0	0	0	2^{-2}
89	0	0	0	e_{30}	e_{29}	0	0	0	0	2^{-2}
90	0	0	e_{30}	e_{29}	0	0	0	0	0	2^{-2}
91	0	e_{30}	e_{29}	0	0	0	0	0	$e_{20}(=\Delta m^{10})$	2^{-3}
92	e_{30}	e_{29}	0	0	0	0	0	e_{20}	0	2^{-3}
93	e_{29}	0	0	0	0	0	e_{20}	e_{19}	0	2^{-3}
94	0	0	0	0	0	e_{20}	e_{19}	e_{18}	0	2^{-3}
95	0	0	0	0	e_{20}	e_{19}	e_{18}	0	$e_{31}(=\Delta m^2)$	2^{-3}
96	0	0	0	e_{20}	e_{19}	e_{18}	0	e_{31}	0	2^{-4}
$0 \sim 95$	$p = 2^{-102}$, $\hat{p} = 2^{-99.1}$ (3-pass HAVAL)									
97	0	0	e_{20}	e_{19}	e_{18}	0	e_{31}	0	0	2^{-4}
98	0	e_{20}	e_{19}	e_{18}	0	e_{31}	0	0	0	2^{-4}
99	e_{20}	e_{19}	e_{18}	0	e_{31}	0	0	0	0	2^{-4}
100	e_{19}	e_{18}	0	e_{31}	0	0	0	e_9	$e_{31}(=\Delta m^2)$	2^{-4}
101	e_{18}	0	e_{31}	0	0	0	e_9	$e_{8,31}$	0	2^{-5}
102	0	e_{31}	0	0	0	e_9	$e_{8,31}$	e_7		
$0 \sim 101$	$p = 2^{-127}$, $\hat{p} = 2^{-124.4}$ (reduced 4-pass HAVAL)									

5.2 Multi-Block Differentials for Rectangle Distinguishers

We now discuss the probability $\hat{p}$ on HMAC using multi-block messages. Assume that two multi-block messages M and M' inserted to H_2 are divided into n block sub-messages $M^1||M^2||\cdots||M^n$ and $M'^1||M'^2||\cdots||M'^n$ with difference $\alpha = \alpha_1\ ||\alpha_2||\cdots||\alpha_n$. Then the initial values for M^i and M'^i are the same as the hash values of the underlying hash function for the sub-messages $M^1||M^2||\cdots||M^{i-1}$ and $M'^1||M'^2||\cdots||M'^{i-1}$ for $2 \leq i \leq n$. Recall that the initial values for the M^1 and M'^1 is the same, namely the output of the compression function for $K \oplus ipad$. Assuming that for the i-th compression function of the H_2 there exist differentials $\beta_{i-1} \rightarrow \beta_i$ with probability p_i under the message difference α_i, where $\beta_0 = 0$, i.e., for a sub-message pair (M^i, M'^i) the input difference β_{i-1} goes to output difference β_i with probability p_i, we get

$$\hat{p} = \sqrt{\sum_{\beta_1,\beta_2,\cdots,\beta_n} (p_1 \times p_2 \times \cdots \times p_n)^2}$$

under the message difference $\alpha_1||\alpha_2||\cdots||\alpha_n$. This is due to the fact that the initial value to the first compression function of the H_2 is not known[6]. So it is much more difficult to apply multi-block messages in our rectangle distinguisher of HMAC compared to the use of one-block messages (in terms of the same number of rounds of compression function). This statement also applies to multi-block differentials for differential distinguishers with the same reasoning. We omit the details of multi-block differentials of HAVAL, MD4, MD5, SHA-0 and SHA-1.

5.3 Differentials for Differential Distinguishers

As stated above, our differential distinguisher works based on a differential which causes a zero difference, i.e., a collision, after the function h_2. We use the foregoing differentials or the previously known differentials on MD4, SHA-0 and SHA-1 in our distinguishing and forgery attacks:

- For SHA-0, the 65-round differential with probability 2^{-78} in Table 5 can be extended into a 82-round differential with probability 2^{-98} ($\approx q$), which causes a collision (this extended differential has appeared in [7], but the differential probability is lower than that in [7] since we cannot use neutral bits.)
- For SHA-1, the first 34-round differential with probability 2^{-52} in Table 6 can be used as our differential distinguisher.
- For the full MD4, there exists a differential with probability 2^{-56} ($\approx q$), which causes a zero output difference from an unknown initial value [26].
- For the full SHA-0, there exists a differential with probability 2^{-107} ($\approx q$), which causes a zero output difference from an unknown initial value [23, 25].

6 Distinguishing and Forgery Attacks on HMAC

We use the probabilities $\hat{p}$ and q to show two distinguishing and a forgery attacks on the HMAC construction, and apply these attacks to HMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1.

Our first distinguishing attack on HMAC using $\hat{p}$ and a rectangle distinguisher is described as follows:

[6] If the initial value is known, the probability $\hat{p}$ is much higher than when it is unknown since the known initial value allows us to find specific sub-messages M_i and M'_i with probability p_i to produce two outputs with difference β_i from $i = 1$ till $i = n$ in order. This method has been introduced in [22, 25, 24, 8].

1. Collect $2^{(l+1)/2} \cdot \hat{p}^{-1}$ message pairs (M_i, M_i') with difference α, where all the M_i and M_i' have the same bit-length t.
2. With a chosen message attack scenario, ask for MAC pairs of all the (M_i, M_i'). We denote the corresponding MAC pairs by (C_i, C_i'). (We assume that the MAC algorithm is either an instantiated HMAC or a random function (or HMAC with a random function) which maps from t bits to l bits.)
3. Check if $C_i \oplus C_j = C_i' \oplus C_j' = 0$ or $C_i \oplus C_j' = C_i' \oplus C_j = 0$ for all i, j such that $1 \leq i < j \leq 2^{(l+1)/2} \cdot \hat{p}^{-1}$. If there is at least one MAC quartet that satisfies this test, output the MAC algorithm = HMAC, otherwise, output the MAC algorithm = a random function (or HMAC with a random function).

The data complexity of this attack is $2^{1+((l+1)/2)} \cdot \hat{p}^{-1}$ chosen messages and this attack requires a memory of $2^{1+(l+1)/2} \cdot \hat{p}^{-1}$ l-bit blocks for storing all the MAC values. The time complexity of this attack is dominated by Step 1 (the data collection time) and Step 3, which seeks colliding MAC quartets. Since it can be done efficiently by sorting the MAC pairs (C_i, C_i')'s by C_i's, the time complexity of this attack is thus a fraction of the time required to compute the MAC values for the chosen messages (Step 1).

We now analyze the success rate of this attack. In Step 1 the $2^{(l+1)/2} \cdot \hat{p}^{-1}$ message pairs form $2^l \cdot \hat{p}^{-2}$ message quartets $((M_i, M_i'),(M_j, M_j'))$ corresponding to MAC quartets $((C_i, C_i'),(C_j, C_j'))$ for $1 \leq i < j \leq 2^{(l+1)/2} \cdot \hat{p}^{-1}$. Since for HMAC $C_i \oplus C_j = C_i' \oplus C_j' = 0$ holds with a probability of $2^{-l} \cdot \hat{p}^2$, and $C_i \oplus C_j' = C_i' \oplus C_j = 0$ also holds with the same probability (this probability has been computed in Sect. 4), the expected number of MAC quartets satisfying the last test is 2 $(= (2^l \cdot \hat{p}^{-2}) \cdot (2^{-l} \cdot \hat{p}^2) + (2^l \cdot \hat{p}^{-2}) \cdot (2^{-l} \cdot \hat{p}^2))$. On the other hand, for a random function (or HMAC with a random function), $C_i \oplus C_j = C_i' \oplus C_j' = 0$ holds with a probability of 2^{-2l}, and $C_i \oplus C_j' = C_i' \oplus C_j = 0$ also holds with the same probability and thus the expectation of satisfying the test is $2^{-l+1} \cdot (\hat{p}^{-2}) (= 2^{-2l} \cdot (2^l \cdot \hat{p}^{-2}) + 2^{-2l} \cdot (2^l \cdot \hat{p}^{-2}))$. Hence, the success rate of this attack is

$$\frac{1 - (1 - 2^{-l} \cdot \hat{p}^2)^{2^{l+1} \cdot \hat{p}^{-2}}}{2} + \frac{(1 - 2^{-2l})^{2^{l+1} \cdot \hat{p}^{-2}}}{2} \approx \frac{1 - e^{-2}}{2} + \frac{e^{-2^{-l+1} \cdot \hat{p}^{-2}}}{2}.$$

Here, the first term is approximately 0.43. Our second distinguishing attack on HMAC using q and a differential distinguisher is described as follows:

1. Collect $2 \cdot q^{-1}$ message pairs (M_i, M_i') with difference α, where all the M_i and M_i' have the same bit-length t.
2. With a chosen message attack scenario, ask for MAC pairs of all the (M_i, M_i'). We denote the corresponding MAC pairs by (C_i, C_i'). We assume that the MAC algorithm is either an instantiated HMAC or a random function (or HMAC with a random function) which maps t bits to l bits.
3. Check if $C_i \oplus C_i' = 0$. If there is at least one MAC pair that satisfies this test, output the MAC algorithm = HMAC, otherwise, output the MAC algorithm = a random function (or HMAC with a random function).

The data complexity of this attack is $2^2 \cdot q^{-1}$ chosen messages and this attack does not require any storage, and the time complexity of this attack itself is a fraction of the time required to compute the MAC values for the chosen messages. Similarly, the success rate of this attack is computed as follows:

$$\frac{1 - (1 - q)^{2 \cdot q^{-1}}}{2} + \frac{(1 - 2^{-l})^{2 \cdot q^{-1}}}{2} \approx \frac{1 - e^{-2}}{2} + \frac{e^{-2^{-l+1} \cdot q^{-1}}}{2}.$$

Finally, our forgery attack on HMAC using q and a differential distinguisher is described as follows:

Table 3. Distinguishing and forgery attacks on HMAC with HAVAL, MD4, MD5, SHA-0 and SHA-1

Hash Function	Type of Distinguisher	Type of Attack	h_2 # of Rounds	Probability of Distinguisher	Data Complexity	Success Rate
3-pass HAVAL (96 rounds)	**R**†	**Distinguishing**	**96 (full)**	$\hat{\mathbf{p}} = \mathbf{2^{-99.1}}$	$\mathbf{2^{228.6}}$	**0.93**
4-pass HAVAL (128 rounds)	R	Distinguishing	102 (reduced)	$\hat{p} = 2^{-124.4}$	$2^{253.9}$	0.93
MD4 (48 rounds)	**R**†	**Distinguishing**	**48 (full)**	$\hat{\mathbf{p}} = \mathbf{2^{-56}}$	$\mathbf{2^{121.5}}$	**0.93**
	D†	**Forgery**	**48 (full)**	$\mathbf{q = 2^{-56}}$	$\mathbf{2^{58}}$	**0.93**
MD5 (64 rounds)	R	Distinguishing	33 (reduced)	$\hat{p} = 2^{-60.6}$	$2^{126.1}$	0.92
SHA-0	R	Distinguishing	65 (reduced)	$\hat{p} = 2^{-78}$	$2^{159.5}$	0.87
	D†	**Distinguishing**	**82 (full)**	$\mathbf{q = 2^{-98}}$	$\mathbf{2^{100}}$	**0.93**
	D†	**Distinguishing**	**80 (full)**	$\mathbf{q = 2^{-107}}$	$\mathbf{2^{109}}$	**0.93**
	D	Forgery	54 (reduced)	$q = 2^{-61}$	2^{63}	0.93
(80 rounds)	D	Forgery	65 (reduced)	$q = 2^{-78}$	2^{80}	0.93
SHA-1	R	Distinguishing	43 (reduced)	$\hat{p} = 2^{-73.4}$	$2^{154.9}$	0.93
(80 rounds)	D	Forgery	34 (reduced)	$q = 2^{-51}$	2^{53}	0.93

†: the attacks can work on HMAC based on full-round (or extended-round) hash functions.
R: Rectangle, D: Differential
Data complexity is the amount of chosen messages
In the rectangle attacks, memory complexity is the same as data complexity
Distinguishing attack is the attack to distinguish instantiated HMAC from HMAC with a random function

1. Run Step 1 in the second distinguishing attack.
2. Run Step 2 in the second distinguishing attack, but we assume that the MAC algorithm is an instantiated HMAC.
3. Check if $C_i \oplus C'_i = 0$ and ask for the MAC pair of $M_i||P$ and $M'_i||P$, where M_i and M'_i have a same MAC value and P is some non-empty string. If the obtained MAC pair collides, again ask for the MAC value of $M_i||P||P'$, where P' is some non-empty string. We denote this obtained MAC value by C. Output C as the MAC value of $M'_i||P||P'$. Otherwise, restart this step until we check all MAC pairs (C_i, C'_i).

It is easy to see that this forgery attack works with (almost) the same data complexity and the same success rate as our second distinguishing attack.

We can easily apply these three attacks to HMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1 by using their probabilities $\hat{p}$ and q. Table 3 shows the results of distinguishing and forgery attacks on those instantiations of HMAC[7]. In Table 3 forgery attacks also imply distinguishing-R and distinguishing-H attacks.

Note: Our distinguishing and forgery attacks are also applicable to HMAC in which the four components h_1, h_2, h_3, h_4 are instantiated with different compression functions (see for example the pseudorandom functions of SSL 3.0). For example, if HMAC employs full-round MD-5, full-round MD-4, full-round MD5 and full-round MD5 for h_1, h_2, h_3 and h_4, respectively, it can be forged with a data complexity of 2^{58} chosen messages. This is due to the fact that our distinguishing and forgery attacks depend only on the function h_2.

[7] These attacks are mounted under the assumption that the output values of the functions h_1 and h_2 distribute uniformly over all possible values when K and M_i are chosen uniformly at random (differential distinguishers are independent of the distributions of the output values of the functions h_2 and H_2).

7 Applications to NMAC

NMAC is a generalized version of HMAC, which uses two l-bit secret keys (K_1, K_2). It is computed as follows:

$$\mathbf{NMAC}(K_1, K_2, M) = H(K_2, H(K_1, M)) .$$

NMAC has exactly the same structure as HMAC except for the use of the keys, i.e., in NMAC the secret keys K_1 and K_2 are used instead of $h_1(IV, K \oplus ipad)$ and $h_3(IV, K \oplus opad)$.

Due to the similar structure, our differential and rectangle distinguishers of HMAC also apply to NMAC. Thus, the distinguishing and forgery attacks described in Sect. 6 also work on NMAC and the results of Table 3 apply NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1.

8 Some Implications of Our Differential and Rectangle Distinguishers

Our differential and rectangle distinguishers in Propositions 1 and 2 can be useful to construct second-preimage attacks on HMAC and NMAC.

It is natural to define a second-preimage resistance for MAC algorithms from that for hash functions:

Second-preimage resistance on MAC algorithms: for any message M, it is computationally infeasible to find another message M' such that $\mathrm{MAC}(K, M') = \mathrm{MAC}(K, M)$, where K is a randomly chosen key.

It follows that for any message M, it should be computationally infeasible for an attacker to find another message M' such that $\mathrm{MAC}(K, M') = \mathrm{MAC}(K, M)$ with a probability larger than 2^{-l}. Since our differential distinguisher uses a probability larger than 2^{-l}, which can find a second preimage with the same differential probability, our differential attacks on HMAC and NMAC imply second-preimage attacks on HMAC.

The second-preimage resistance on MAC algorithms also follows that for any message pair (M_i, M'_j), an attacker cannot find another message pair (M'_i, M_j) such that $\mathrm{MAC}(K, M_i) = \mathrm{MAC}(K, M_j)$ and $\mathrm{MAC}(K, M'_i) = \mathrm{MAC}(K, M'_j)$ with a probability larger than 2^{-2l}. This implies that our rectangle distinguisher is applicable to second-preimage attacks on HMAC and NMAC. For example, consider the distinguishing-H attack on HMAC-HAVAL(3-pass) in Table 3. From the probability $\hat{p} = 2^{-99.1}$, we know that for a given message pair (M_i, M'_j), our rectangle distinguisher can find another message pair (M'_i, M_j) such that $f(K, M_i) = f(K, M_j)$ and $f(K, M'_i) = f(K, M'_j)$ with a probability of approximately $(2^{-99.1})^2 \cdot 2^{-256} = 2^{-452.2}$ which is much larger than $(2^{-256})^2 = 2^{-512}$, where f =HMAC-HAVAL(3-pass) (refer to Fig. 2. and Table 2).

The second-preimage resistance on MAC algorithms is a weakened security notion of forgery. Indeed, a second-preimage attack implies a forgery. However, the converse dose not hold since second-preimage attacks are first given a target message. This security notion is also very important if meaningful messages are considered.

9 Conclusions

We have presented differential and rectangle distinguishers on HMAC, which are derived from its structural property. They allow to present distinguishing and forgery attacks on HMAC that can be mounted when HMAC employs hash functions with slow difference propagations. With these distinguishing and forgery attacks we have shown that HMAC with the

full versions of 3-pass HAVAL and SHA-0 can be distinguished from HMAC with a random function, and HMAC with the full version of MD4 can be forged. These distinguishing and forgery attacks have also been applied to HMAC based on reduced versions of MD5 and SHA-1. We have also shown that our distinguishing and forgery attacks can be mounted on NMAC (which is a generalized version of HMAC) with the same complexity. Furthermore, we have shown that our differential and rectangle distinguishers can lead to second-preimage attacks on HMAC and NMAC. All these attacks do not contradict the security proof of HMAC, but they improve our understanding of the security of HMAC based on existing cryptographic hash functions.

Our differential distinguisher on HMAC works only if the underlying hash function has a differential with a zero output difference with probability larger than $2^{-|\text{hash value}|}$. Our rectangle distinguisher on HMAC works only if the underlying hash function has differentials such that the sum of the square of their probabilities is larger than $2^{-|\text{hash value}|}$. Unlike the previous attacks on hash functions, our analysis on the hash function embedded in HMAC should be done under an unknown fixed initial value (which is determined by a secret key). This fact makes difficult to use the recently proposed message modification technique (Wang et al.'s attacks) and neutral-bit technique (Biham et al.'s attacks) in analyzing HMAC based on specific cryptographic hash functions. However, it is interesting to investigate if their methods can be applied to HMAC with some new other techniques when HMAC is instantiated with a specific cryptographic hash function. We expect that the method developed in this paper would be useful for the further analysis of HMAC.

References

1. M. Bellare, *New Proofs for NMAC and HMAC: Security without Collision-Resistance*, Advances in Cryptology – Proceedings of CRYPTO 2006, to appear, and Cryptology ePrint Archive, Report 2006/043, Available Online at *http://eprint.iacr.org/2006/043.pdf*
2. M. Bellare, R. Canetti and H. Krawczyk, *Keying Hash Functions for Message Authentication*, Advances in Cryptology – Proceedings of CRYPTO 1996, LNCS 1109, pp. 1-15, Springer-Verlag, 1996.
3. M. Bellare, J. Kilian and P. Rogaway, *The Security of the Cipher Block Chaining Message Authentication Code*, Journal of Computer and System Sciences, Vol. 61, No. 3, pp. 362-399, Dec 2000.
4. M. Bellare, P. Rogaway, and D. Wagner, *The EAX Mode of Operation*, Proceedings of FSE 2004, LNCS 3017, pp. 389-407, Springer-Verlag, 2004.
5. D.J. Bernstein, *The Poly1305-AES Message-Authentication Code*, Proceedings of FSE 2005, to appear.
6. E. Biham, O. Dunkelman and N. Keller, *The Rectangle Attack – Rectangling the Serpent*, Advances in Cryptology – Proceedings of EUROCRYPT 2001, LNCS 2045, pp. 340-357, Springer-Verlag, 2001.
7. E. Biham and R. Chen, *Near-Collisions of SHA-0*, Advances in Cryptology – Proceedings of CRYPTO 2004, LNCS 3152, pp. 290-305, Springer-Verlag, 2004.
8. E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet and W. Jalby, *Collisions of SHA-0 and Reduced SHA-1*, Advances in Cryptology – Proceedings of EUROCRYPT 2005, LNCS 3494, pp. 22-35, Springer-Verlag, 2005.
9. J. Black, S. Halevi, H. Krawczyk, T. Krovetz and P. Rogaway, *UMAC: fast and secure message authentication*, Advances in Cryptology – Proceedings of CRYPTO 1999, LNCS 1666, pp. 216-233, Springer-Verlag, 1999.
10. F. Chabaud and A. Joux, *Differential Collisions in SHA-0*, Advances in Cryptology – Proceedings of CRYPTO 1998, LNCS 1462, pp. 56-71, Springer-Verlag, 1999.
11. ISO/IEC 9797, Data Cryptographic Techniques - Data Integrity Mechanism Using a Cryptographic Check Function Employing a Block Cipher Algorithm, 1989.
12. T. Iwata and K. Kurosawa, *OMAC: One-Key CBC MAC*, Proceedings of FSE 2003, LNCS 2887, pp. 129-153. Springer-Verlag, 2003.

13. E. Jaulmes, A. Joux and F. Valette, *On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction*, Proceedings of FSE 2002, LNCS 2365, pp. 237-251, Springer-Verlag, 2002.
14. T. Kohno, J. Viega and D. Whiting, *CWC: A High-Performance Conventional Authenticated Encryption Mode*, Proceedings of FSE 2004, LNCS 3017, pp. 408-426, Springer-Verlag, 2004.
15. K. Kurosawa and T. Iwata, *TMAC: Two-Key CBC MAC. Topics in Cryptology*, Proceedings of CT-RSA 2003, LNCS 2612, pp. 33-49, Springer-Verlag, 2003.
16. D.A. McGrew and J. Viega, *The Galois/Counter Mode of Operation (GCM)*, http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm-spec.pdf
17. B. Preneel, A. Bosselaers, R. Govaerts and J. Vandewalle, *A chosen text attack on the modified cryptographic checksum algorithm of Cohen and Huang*, Advances in Cryptology – Proceedings of CRYPTO 1989, LNCS 435, pp. 154-163, Springer-Verlag, 1990.
18. B. Preneel, P.C. van Oorschot, *MDx-MAC and building fast MACs from hash functions*, Advances in Cryptology – Proceedings of CRYPTO 1995, LNCS 963, pp. 1-14, Springer-Verlag, 1995.
19. R.L. Rivest, *The MD4 Message Digest Algorithm*, Advances in Cryptology – Proceedings of CRYPTO 1990, LNCS 537, pp. 303-311, Springer-Verlag, 1991.
20. R.L. Rivest, *The MD5 Message Digest Algorithm*, Request for Comments (RFC 1320), Internet Activities Board, Internet Privacy Task Force, 1992.
21. G. Rose, *A Stream Cipher Based on Linear Feedback over* $GF(2^8)$, Proceedings of ACISP 1998, LNCS 1438, pp. 135-146, Springer-Verlag, 1998.
22. X. Wang and H. Yu, *How to Break MD5 and Other Hash Functions*, Advances in Cryptology – Proceedings of EUROCRYPT 2005, LNCS 3494, pp. 19-35, Springer-Verlag, 2005.
23. X. Wang, X. Lai, D. Feng, H. Chen and X. Yu, *Cryptanalysis of the Hash Functions MD4 and RIPEMD*, Advances in Cryptology – Proceedings of EUROCRYPT 2005, LNCS 3494, pp. 1-18, Springer-Verlag, 2005.
24. X. Wang, Y.L. Yin and H. Yu, *Finding Collisions in the Full SHA-1*, Advances in Cryptology – Proceedings of CRYPTO 2005, LNCS 3621, pp. 17-36, Springer-Verlag, 2005.
25. X. Wang, H. Yu and Y.L. Yin, *Efficient Collision Search Attacks on SHA-0*, Advances in Cryptology – Proceedings of CRYPTO 2005, LNCS 3621, pp. 1-16, Springer-Verlag, 2005.
26. H. Yu, G. Wang, G. Zhang and X. Wang, *The Second-Preimage Attack on MD4*, Proceedings of CANS 2005, LNCS 3810, pp. 1-12, Springer-Verlag, 2005.
27. Y. Zheng, J. Pieprzyk and J. Seberry, *HAVAL-A One-way Hashing Algorithm with Variable Length of Output*, Advances in Cryptology – Proceedings of AUSCRYPT 1992, LNCS 718, pp. 83-104, Springer-Verlag, 1993.
28. U.S. Department of Commerce. *FIPS 180*: Secure Hash Standard, Federal Information Processing Standards Publication, N.I.S.T., May 1993.
29. U.S. Department of Commerce. *FIPS 180-1*: Secure Hash Standard, Federal Information Processing Standards Publication, N.I.S.T., April 1995.

A Differentials on the Compression Functions of MD5, SHA-0 and SHA-1

Table 4. A Differential for Rounds $0 \sim 32$ on MD5

Round (i)	ΔA^i	ΔB^i	ΔC^i	ΔD^i	Δm^i	Prob.
0	0	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
7	0	0	0	0	0	1
8	0	0	0	0	e_{24}	2^{-1}
9	0	e_{31}	0	0	e_{19}	2^{-2}
10	0	0	e_{31}	0	0	2^{-1}
11	0	0	0	e_{31}	0	2^{-1}
12	e_{31}	0	0	0	e_{31}	1
13	0	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
23	0	0	0	0	0	1
24	0	0	0	0	e_{19}	2^{-2}
25	0	e_{24}	0	0	0	2^{-2}
26	0	e_{24}	e_{24}	0	0	2^{-3}
27	0	$e_{6,24}$	e_{24}	e_{24}	e_{24}	2^{-5}
28	e_{24}	$e_{6,24}$	$e_{6,24}$	e_{24}	0	2^{-6}
29	e_{24}	$e_{6,11,24}$	$e_{6,24}$	$e_{6,24}$	0	2^{-7}
30	$e_{6,24}$	$e_{6,11,24}$	$e_{6,11,24}$	$e_{6,24}$	0	2^{-9}
31	$e_{6,24}$	$e_{6,11,24}$	$e_{6,11,24}$	$e_{6,11,24}$	e_{31}	2^{-9}
32	$e_{6,11,24}$	$e_{6,11,19,24}$	$e_{6,11,24}$	$e_{6,11,24}$	0	2^{-8}
33	$e_{6,11,24}$	$e_{6,11,23}$	$e_{6,11,19,24}$	$e_{6,11,24}$		
$0 \sim 32$	$p = 2^{-69}$, $\hat{p} = 2^{-60.6}$					

Table 5. A Differential on SHA-0

Round (i)	ΔA^i	ΔB^i	ΔC^i	ΔD^i	ΔE^i	Δm^i	Prob.
0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1
3	0	0	0	0	0	e_1	2^{-1}
4	e_1	0	0	0	0	e_6	2^{-1}
5	0	e_1	0	0	0	e_1	2^{-2}
6	0	0	e_{31}	0	0	e_{31}	2^{-1}
7	0	0	0	e_{31}	0	e_{31}	2^{-1}
8	0	0	0	0	e_{31}	$e_{1,31}$	2^{-1}
9	e_1	0	0	0	0	e_6	2^{-1}
10	0	e_1	0	0	0	0	2^{-2}
11	e_1	0	e_{31}	0	0	$e_{6,31}$	2^{-2}
12	0	e_1	0	e_{31}	0	$e_{1,31}$	2^{-3}
13	0	0	e_{31}	0	e_{31}	e_1	2^{-2}
14	e_1	0	0	e_{31}	0	$e_{6,31}$	2^{-2}
15	0	e_1	0	0	e_{31}	$e_{1,31}$	2^{-2}
16	0	0	e_{31}	0	0	e_{31}	2^{-1}
17	0	0	0	e_{31}	0	$e_{1,31}$	2^{-2}
18	e_1	0	0	0	e_{31}	$e_{1,6,31}$	2^{-2}
19	e_1	e_1	0	0	0	e_6	2^{-3}
20	e_1	e_1	e_{31}	0	0	$e_{6,31}$	2^{-2}
21	e_1	e_1	e_{31}	e_{31}	0	$e_{1,6}$	2^{-2}
22	0	e_1	e_{31}	e_{31}	e_{31}	$e_{1,31}$	2^{-1}
23	0	0	e_{31}	e_{31}	e_{31}	$e_{1,31}$	2^{-1}
24	e_1	0	0	e_{31}	e_{31}	e_6	2^{-1}
25	0	e_1	0	0	e_{31}	e_{31}	2^{-1}
26	e_1	0	e_{31}	0	0	$e_{1,6,31}$	2^{-2}
27	e_1	e_1	0	e_{31}	0	$e_{1,6,31}$	2^{-2}
28	0	e_1	e_{31}	0	e_{31}	e_1	2^{-1}
29	0	0	e_{31}	e_{31}	0	0	1
30	0	0	0	e_{31}	e_{31}	0	1
31	0	0	0	0	e_{31}	e_{31}	1
32	0	0	0	0	0	e_1	2^{-1}
33	e_1	0	0	0	0	$e_{1,6}$	2^{-2}
34	e_1	e_1	0	0	0	e_6	2^{-2}
35	e_1	e_1	e_{31}	0	0	$e_{1,6,31}$	2^{-2}
36	0	e_1	e_{31}	e_{31}	0	e_1	2^{-1}
37	0	0	e_{31}	e_{31}	e_{31}	e_{31}	1
38	0	0	0	e_{31}	e_{31}	0	1
39	0	0	0	0	e_{31}	e_{31}	1
40	0	0	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
45	0	0	0	0	0	0	1
46	0	0	0	0	0	e_1	2^{-1}
47	e_1	0	0	0	0	$e_{1,6}$	2^{-2}
48	e_1	e_1	0	0	0	$e_{1,6}$	2^{-3}
49	0	e_1	e_{31}	0	0	$e_{1,31}$	2^{-3}
50	0	0	e_{31}	e_{31}	0	0	2^{-1}
51	0	0	0	e_{31}	e_{31}	0	2^{-1}
52	0	0	0	0	e_{31}	e_{31}	1
53	0	0	0	0	0	0	1
54	0	0	0	0	0	e_1	2^{-1}
55	e_1	0	0	0	0	$e_{1,6}$	2^{-2}
56	e_1	e_1	0	0	0	$e_{1,6}$	2^{-3}
57	0	e_1	e_{31}	0	0	e_{31}	2^{-3}
58	e_1	0	e_{31}	e_{31}	0	$e_{1,6}$	2^{-3}
59	e_1	e_1	0	e_{31}	e_{31}	$e_{1,6}$	2^{-4}
60	0	e_1	e_{31}	0	e_{31}	e_1	2^{-1}
61	0	0	e_{31}	e_{31}	0	0	1
62	0	0	0	e_{31}	e_{31}	0	1
63	0	0	0	0	e_{31}	e_{31}	1
64	0	0	0	0	0	0	1
65	0	0	0	0	0	e_1	2^{-1}
$0 \sim 64$	$p = 2^{-78}$, $\hat{p} = 2^{-78}$						
66	e_1	0	0	0	0	$e_{1,6}$	2^{-2}
67	e_1	e_1	0	0	0	$e_{1,6}$	2^{-2}
68	0	e_1	e_{31}	0	0	$e_{1,31}$	2^{-1}
69	0	0	e_{31}	e_{31}	0	0	1
70	0	0	0	e_{31}	e_{31}	e_1	2^{-1}
71	e_1	0	0	0	e_{31}	$e_{6,31}$	2^{-1}
72	0	e_1	0	0	0	0	2^{-1}
73	e_1	0	e_{31}	0	0	$e_{1,6,31}$	2^{-2}
74	e_1	e_1	0	e_{31}	0	$e_{1,6,31}$	2^{-2}
75	0	e_1	e_{31}	0	e_{31}	0	2^{-1}
76	e_1	0	e_{31}	e_{31}	0	$e_{1,6}$	2^{-2}
77	e_1	e_1	0	e_{31}	e_{31}	$e_{1,6}$	2^{-2}
78	0	e_1	e_{31}	0	e_{31}	e_1	2^{-1}
79	0	0	e_{31}	e_{31}	0	0	1
80	0	0	0	e_{31}	e_{31}	0	2^{-1}
81	0	0	0	0	e_{31}	e_{31}	1
82	0	0	0	0	0		
$0 \sim 81$	$q = 2^{-98}$						

Table 6. A Differential for Rounds $0 \sim 42$ on SHA-1

Round (i)	ΔA^i	ΔB^i	ΔC^i	ΔD^i	ΔE^i	Δm^i	Prob.
0	0	0	0	0	0	e_1	2^{-1}
1	e_1	0	0	0	0	e_6	1
2	0	e_1	0	0	0	0	2^{-2}
3	e_1	0	e_{31}	0	0	$e_{6,31}$	2^{-2}
4	0	e_1	0	e_{31}	0	e_{31}	2^{-3}
5	e_1	0	e_{31}	0	e_{31}	e_6	2^{-2}
6	0	e_1	0	e_{31}	0	$e_{0,31}$	2^{-4}
7	$e_{0,1}$	0	e_{31}	0	e_{31}	$e_{5,6}$	2^{-3}
8	0	$e_{0,1}$	0	e_{31}	0	$e_{0,1,31}$	2^{-5}
9	0	0	$e_{30,31}$	0	e_{31}	$e_{1,30}$	2^{-4}
10	e_1	0	0	$e_{30,31}$	0	$e_{6,30,31}$	2^{-4}
11	0	e_1	0	0	$e_{30,31}$	$e_{1,30,31}$	2^{-3}
12	0	0	e_{31}	0	0	e_{31}	2^{-1}
13	0	0	0	e_{31}	0	e_{31}	2^{-1}
14	0	0	0	0	e_{31}	$e_{1,31}$	2^{-1}
15	e_1	0	0	0	0	e_6	2^{-1}
16	0	e_1	0	0	0	e_1	2^{-2}
17	0	0	e_{31}	0	0	e_{31}	2^{-1}
18	0	0	0	e_{31}	0	e_{31}	2^{-1}
19	0	0	0	0	e_{31}	e_{31}	1
20	0	0	0	0	0	e_1	2^{-1}
21	e_1	0	0	0	0	e_6	2^{-1}
22	0	e_1	0	0	0	0	2^{-1}
23	e_1	0	e_{31}	0	0	$e_{6,31}$	2^{-1}
24	0	e_1	0	e_{31}	0	$e_{1,31}$	2^{-1}
25	0	0	e_{31}	0	e_{31}	0	1
26	0	0	0	e_{31}	0	e_{31}	1
27	0	0	0	0	e_{31}	e_{31}	1
28	0	0	0	0	0	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
33	0	0	0	0	0	0	1
34	0	0	0	0	0	e_2	2^{-1}
35	e_2	0	0	0	0	e_7	2^{-1}
36	0	e_2	0	0	0	e_2	2^{-1}
37	0	0	e_0	0	0	$e_{0,3}$	2^{-2}
38	e_3	0	0	e_0	0	$e_{0,2,8}$	2^{-3}
39	e_2	e_3	0	0	e_0	$e_{0,3,7}$	2^{-3}
40	0	e_2	e_1	0	0	$e_{1,2,4}$	2^{-5}
41	e_4	0	e_0	e_1	0	$e_{0,1,3,9}$	2^{-6}
42	e_3	e_4	0	e_0	e_1	$e_{0,1,3,4,8}$	2^{-7}
43	e_3	e_3	e_2	0	e_0		
$0 \sim 42$	$p = 2^{-75}$, $\hat{p} = 2^{-73.4}$						