Journal of Universal Computer Science, vol. 14, no. 3 (2008), 347-376
submitted: 2/6/07, accepted: 1/11/08, appeared: 1/2/08

New Results on NMAC/HMAC when Instantiated with Popular Hash Functions

Christian Rechberger and Vincent Rijmen
(Institute for Applied Information Processing and Communications, Graz University of Technology, Austria
christian.rechberger, vincent.rijmen@iaik.tugraz.at)

Abstract: Message Authentication Code (MAC) algorithms can provide cryptographically secure authentication services. One of the most popular algorithms in commercial applications is HMAC based on the hash functions MD5 or SHA-1. In the light of new collision search methods for members of the MD4 family including SHA-1, the security of HMAC based on these hash functions is reconsidered.

We present a new method to recover both the inner- and the outer key used in HMAC when instantiated with a concrete hash function by observing text/MAC pairs. In addition to collisions, also other non-random properties of the hash function are used in this new attack. Among the examples of the proposed method, the first theoretical full key recovery attack on NMAC-MD5 is presented. Other examples are distinguishing, forgery and partial or full key recovery attacks on NMAC/HMAC-SHA-1 with a reduced number of steps (up to 62 out of 80). This information about the new, reduced security margin serves as an input to the selection of algorithms for authentication purposes.

Key Words: cryptography, security, authentication
Category: C.2.0, D.4.6, E.3, K.6.5

1 Introduction

Authentication services can be provided in a cryptographically secure way by using Message Authentication Codes (MACs). The two most popular algorithms in commercial applications are variants of CBC-MAC based on 3-DES or AES, and HMAC based on MD5 and SHA-1. NMAC and HMAC [Bellare et al. 1996] are message authentication codes based on a hash function. HMAC has been included in standards like ANSI, IETF, ISO, or FIPS. Commercial applications often use HMAC with SHA-1 as underlying hash function. After the recent successful collision attacks on MD5 [Wang et al. 2005b] and reduced versions of SHA-1 [De Cannière et al. 2007, De Cannière and Rechberger 2006, Wang et al. 2005a], the impact of new collision attacks on the security of MAC constructions needs to be considered. This issue was already briefly mentioned in [Yu et al. 2005] and the impact on early hash based MAC constructions like the prefix-, suffix-, or envelope-method was informally discussed in [Wang 2005]. Of particular interest is the impact on HMAC, since *e. g.* NIST supports HMAC-SHA-1 even after 2010 [NIST 2006], whereas support for SHA-1 as a hash function will be dropped. In addition HMAC-SHA-1 continues to be used in new designs and applications, *e. g.* in [Raihi et al. 2005].

The security proof of NMAC, HMAC and other constructions is based on some assumptions about the pseudo-randomness of the underlying hash function, concluding that collision resistance is not needed [Bellare 2006]. On the other hand, earlier work [Contini and Yin 2006, Kim et al. 2006] suggests that collision attacks on the underlying hash function can be used to weaken the security of HMAC when instantiated with a popular hash function. In particular, distinguishing and forgery attacks, but as yet no full key recovery attacks have been shown[1]

After some preliminaries in Section 2 and introducing some terminology and technicalities related to probabilities in Section 3, we present the two key points of this paper which are as follows.

1. Previous work on the security of NMAC and HMAC [Contini and Yin 2006, Kim et al. 2006] against differential attacks is based on the reuse of characteristics that were constructed in order to mount collision attacks on the underlying hash function. This can lead to optimistic conclusions on the security margin of NMAC and HMAC. We propose a general framework for classifying the non-random properties of compression functions in Section 4. In addition to putting the existing characteristics for MD5, SHA-0 and SHA-1 into this framework, we devise new characteristics suitable for more efficient attacks than previously known on NMAC/HMAC instantiated with reduced variants of SHA-1.

2. The ability to recover the secret key by using known text/MAC pairs is certainly the most dangerous attack in practice. Currently known methods for NMAC/HMAC only allow to recover an inner key. This allows forgery attacks but does not give an attacker the same possibilities as having the key (or equivalent information).

   We show a new key recovery attack which can recover the full key of NMAC (or an equivalent information in the case of HMAC), hence we have for the first time the potential to use a substantially smaller amount of text/MAC pairs than black-box attacks. The details depend on the hash function being used and are discussed in Section 5. There we also give examples for full MD5 and reduced SHA-1.

We summarize and discuss the impact of our results and the security margins offered by HMAC when instantiated with popular hash functions in Section 6. Conclusions and open problems are given in Section 7.

[1] After the presentation of a preliminary version [Rechberger and Rijmen 2007] of our results described in this paper, attacks on HMAC-MD4 and parts of our result on NMAC-MD5 were independently obtained in [Fouque et al. 2007]. See Table 6 for a comparison.

2 Preliminaries

NMAC and HMAC are constructions that instantiate hash functions as their main building blocks. In order to give all the preliminaries, we proceed as follows. First we introduce the main design principle for iterated hash functions, then we describe SHA-1 as an instantiation of it. At the end, we describe NMAC and HMAC.

2.1 Iterated hash functions - the Damgård-Merkle principle

A hash function is expected to accept almost arbitrary long message inputs. Construction method like the Damgård [Damgård 1989]-Merkle [Merkle 1989] iteration principle utilize a *compression function* which operated on inputs of fixed length: Let $f(h_i, m_i)$ denote such a compression function accepting as input a message block m_i and a chaining input h_i where h_0 is a pre-specified initial value. All message blocks are iteratively processed by the iteration $h_{i+1} = f(h_i, m_i)$. Before the input message is separated into blocks, its length is appended and padded to be a multiple of the message block length. The hash value of the N block message M (including padding) under the hash function $H(M)$ is hence defined to be h_N. Most hash functions in use today use this iteration principle.

2.2 The hash function SHA-1

The hash function SHA-1 was designed by the US National Security Agency (NSA) and adopted as a standard in 1995, is widely used, and is representative for a large class of hash functions which started with MD4 and MD5 and includes most algorithms in use today.

SHA-1 is an iterative hash function that processes 512-bit input message blocks and produces a 160-bit hash value, *cf.* [NIST 2002]. Also SHA-1 employs the Damgård-Merkle iteration principle as outlined above. The compression function processes input message blocks of 512 bits and produces a 160-bit chaining value. The compression function of SHA-1 basically consists of two parts: the message expansion and the state update transformation. In the following, we will describe both parts in detail.

2.3 SHA-1 message expansion

In SHA-1, the message expansion is defined as follows. A single 512-bit input message block is represented by 16 32-bit words, denoted by M_i, with $0 \leq i \leq 15$. The message input is linearly expanded into 80 32-bit words W_i defined as follows:

$$W_i = \begin{cases} M_i & \text{for } i < 16\,, \\ (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1\,. \end{cases}$$

2.4 SHA-1 state update transformation

The state update transformation starts from a fixed initial value IV for 5 32-bit registers $A, B, \ldots, E$ (also referred to as state variables) and updates these registers in 80 steps ($i = 0, \ldots, 79$) by using the word W_i and the step constant K_i in step i. One step of the state update transformation is defined as:

$$\begin{aligned}
A_{i+1} &= A_i \lll 5 + W_i + f(B_i, C_i, D_i) + K_i \\
B_{i+1} &= A_i \\
C_{i+1} &= B_i \ggg 2 \\
D_{i+1} &= C_i \\
E_{i+1} &= D_i
\end{aligned}$$

Note that $B_i = A_{i-1}$, $C_i = A_{i-2} \ggg 2$, $D_i = A_{i-3} \ggg 2$, $E_i = A_{i-4} \ggg 2$. This also implies that the chaining inputs fill all A_j for $-4 \leq j \leq 0$. Thus it suffices to consider the state variable A, which we will for the remainder of this article. The function f depends on the step number: steps $i = 0, \ldots, 19$ use the *IF-function* referred to as f_{IF} and steps $i = 40, \ldots, 59$ use the *MAJ-function* referred to as f_{MAJ}. The remaining steps use a 3-input XOR referred to as f_{XOR}. The Boolean functions are defined as follows:

$$\begin{aligned}
f_{IF}(B, C, D) &= (B \wedge C) \oplus (\neg B \wedge D) \\
f_{MAJ}(B, C, D) &= (B \wedge C) \oplus (B \wedge D) \oplus (C \wedge D) \\
f_{XOR}(B, C, D) &= B \oplus C \oplus D\ ,
\end{aligned}$$

where $\wedge$ and $\neg$ denote the bitwise AND operation and the bitwise complement. The step constants K_i and initial values can be found in [NIST 2002]. We do not list them in this section since they do not have any impact on our analysis.

After the last step of the state update transformation, the chaining variables $A_0, A_{-1}, A_{-2}, A_{-3}, A_{-4}$ and the output values of the last step $A_{80}, A_{79}, A_{78}, A_{77}$, A_{76} are combined, resulting in the final value of one iteration (feed forward). The result is the final hash value or the initial value for the next message block.

2.5 NMAC and HMAC

Message authentication codes are mappings that have a secret key and a message as input and produce as output a *tag*. Let $h(iv, m)$ denote the application of an iterative hash function h on message input m and with the chaining variable initialized to iv. The NMAC construction can then be described as follows:

$$NMAC(k_1, k_2, m) = h(k_2, h(k_1, m)). \tag{1}$$

We call the key k_1 and the corresponding h the *inner key* and the *inner hash*. Likewise k_2 is referred to as *outer key*.

HMAC is very similar to NMAC, but reduces k_1 and k_2 to a single key k by means of a key-derivation function which is again the compression function of the hash function.

$$k_1 = h(iv, k \oplus ipad) \quad (2)$$

$$k_2 = h(iv, k \oplus opad) \quad (3)$$

$$HMAC(k, m) = h(k_2, h(k_1, m)). \quad (4)$$

3 Characteristics and probabilities

3.1 Terminology from differential cryptanalysis

Differential cryptanalysis was originally invented to attack DES and other block ciphers [Biham and Shamir 1991]. The key concept behind a differential attack is the definition of a *characteristic.* Considering two inputs to the same cryptographic primitive, a characteristic is defined as the sequence of differences between the intermediate results occurring at corresponding times during the processing of these two inputs. The power of the method lies in the fact that it is possible to predict the differences of intermediate variables without specifying the actual input values. For linear functions, the output difference is fully determined by the input difference. For nonlinear functions, it is possible to predict the output difference with a certain probability.

The *probability of a characteristic* is defined as the fraction of the input pairs that exhibits the differences of the characteristic. These input pairs are called the *right pairs.* In a differential attack, the cryptanalyst first tries to define a characteristic with a high probability. Subsequently, the cryptanalyst searches for one or more right pairs. The complexity of the search is related to the probability of the characteristic, but there are some fine points to consider. The most important characteristics are those it is the easiest to find a right pair for.

In order to be of use in a collision attack on a hash function, a characteristic needs to result in output difference zero. In key recovery attacks also other characteristics can be of use, provided that their probability is high enough.

In a *related-key differential attack*, also characteristics with differences in the key input of the cryptographic primitive are allowed. This often allows to construct characteristics with a higher probability, but the attack scenario becomes less realistic.

3.2 Easy relations

One approach to estimate the complexity of the search for a message pair is to count the total number of conditions, as is done by Kim *et al.* [Kim et al. 2006].

However, when the message is under full control of the attacker, optimizations are possible. It was already observed [Chabaud and Joux 1998] in the early analysis of SHA that some conditions can be expressed as linear relations between the message bits. When considering only messages that satisfy these relations, the probability of a characteristic increases. In the remainder of this paper, we will unless noted otherwise quote the increased probability of a characteristic, *i. e.* for messages that satisfy these linear relations between message bits (easy relations). We will use $\mathcal{M}$ to express this set of easy relations. The increase is significant, as can be illustrated by considering the different step transformations in the compression function of SHA-1.

3.3 Disturbances and local collisions

Characteristics for SHA-1 can be built up of disturbances and their corresponding local collisions. Local collisions as an analysis tool where already used by Chabaud and Joux in their original analysis of SHA [Chabaud and Joux 1998], the predecessor of SHA-1.

Their probability and hence their contribution to the data complexity of attacks devised later on in the paper depends on the bit position in the word and the steps they cover. The reason is that the 3-input Boolean function f being used in the step transformation of SHA-1 changes with every round (group of 20 steps). Table 1 illustrates the different cases. The column 'total' refers to the number of conditions in the case where no relations between message bits are assumed. For all steps and for all bit positions it is possible to *improve the results by fixing some relations between message bits* (shown in column 'reduced'). The number of linear relations between message bits is actually the difference between both numbers. Note that Table 1 is simplified in the sense that local collisions at the border between rounds are not considered. A more comprehensive table is given in the Appendix in Tables 14 and 15. Also note that the position of local collisions relative to each other can either improve the overall probability or lead to impossible differentials. Examples of these effects are in detail discussed in [Mendel et al. 2006]. In the attacks presented in this paper, these effects on the probability of the characteristics are taken into consideration.

3.4 Example for a characteristic

To illustrate this with an example, we can increase the probability of the 34-step SHA-1 characteristic as presented in [Kim et al. 2006] from 2^{-52} to 2^{-31}. See Table 2 for details. For the characteristics, we adopt the notation introduced in [De Cannière and Rechberger 2006]. Here we briefly restate the relevant parts. 'x ' denotes XOR difference of unknown sign, 'n ' and 'u ' denote differences of known sign, '- ' refers to no difference and '1 ' and '0 ' refer to a setting where

Table 1: Number of conditions for a local collision. Note that the given figures only hold if the five steps after the disturbance are within the same round.

bit position	function	total	reduced
$0, 2, \ldots, 25, 27, \ldots, 30$	f_{IF}	9	5
1	f_{IF}	6	5
26	f_{IF}	8	5
31	f_{IF}	7	4
$0, 2, \ldots, 25, 27, \ldots, 30$	f_{XOR}	6	4
1	f_{XOR}	3	2
26	f_{XOR}	5	4
31	f_{XOR}	4	3
$0, 2, \ldots, 25, 27, \ldots, 30$	f_{MAJ}	9	4
1	f_{MAJ}	6	4
26	f_{MAJ}	8	4
31	f_{MAJ}	7	4

not only there is no difference, but also the actual value for the bit is fixed. Column A_i shows the state variables and W_i the expanded message words. The values in the column $P_u(i)$ denote $-log_2(p_u(i))$, where $p_u(i)$ is the uncontrolled probability as defined in [De Cannière and Rechberger 2006].

3.5 Multiple characteristics in one differential

In situations where only the differences at the inputs/outputs matter, but the exact sequence of differences through the computation do not, the concept of *differential* becomes important. Informally, all characteristics with the same input/output difference contribute to the probability of a differential.

To obtain better estimates for the complexity of the search phase, we can add up the probabilities of all characteristics that contribute to the same differential. Mendel *et al.* derive an analytical formula taking into account the effect of multiple characteristics based on a study of carry effects [Mendel et al. 2006]. Tables 14 and 15 give an overview for the case of *independent* local collisions. As an example consider the forgery attack on 37-step HMAC-SHA-1 given in Table 5. By considering the better lower bounds using the methods described above, we expect the forgery attack to be successful already after about 2^{65} instead of 2^{67} chosen messages, since the probability of the used differential is a factor 4 higher than the probability of the best characteristic.

Table 2: Type 2 characteristic with probability 2^{-31}.

i	∇A_i	∇W_i	$P_c(i)$
-4	--------------------------------		
-3	--------------------------------		
-2	--------------------------------		
-1	--------------------------------		
0	--------------------------------	n-------------------------------	1
1	u-------------------------------	---------------------------n----	1
2	--1-----------------------------	--------------------------------	3
3	n-0-----------------------------	--n------------------------u----	1
4	--1-----------------------------	--n-----------------------------	4
5	u-0----------------------------1	---------------------------n----	2
6	--1----------------------------0	-uu-----------------------------	3
7	-n0-----------------------------	---------------------------un---	1
8	---1----------------------------	uun-----------------------------	1
9	---0----------------------------	u--n----------------------------	1
10	u-------------------------------	--un-----------------------n----	1
11	--1-----------------------------	u-un----------------------------	2
12	--0-----------------------------	--n-----------------------------	0
13	--------------------------------	--n-----------------------------	0
14	--------------------------------	n-n-----------------------------	1
15	n-------------------------------	---------------------------u----	1
16	--1-----------------------------	n-------------------------------	2
17	--0-----------------------------	--u-----------------------------	0
18	--------------------------------	--u-----------------------------	0
19	--------------------------------	--u-----------------------------	0
20	--------------------------------	n0------------------------------	1
21	n-------------------------------	---------------------------u----	0
22	--------------------------------	-0------------------------------	1
23	n-------------------------------	--u------------------------u----	1
24	--------------------------------	n-u-----------------------------	1
25	--------------------------------	--------------------------------	1
26	--------------------------------	-1n-----------------------------	1
27	--------------------------------	-1u-----------------------------	0
28	--------------------------------	--------------------------------	0
29	--------------------------------	--------------------------------	0
30	--------------------------------	--------------------------------	0
31	--------------------------------	--------------------------------	0
32	--------------------------------	--------------------------------	0
33	--------------------------------	--------------------------------	0
34	--------------------------------		

4 New characteristics

In this section, we first categorize the known characteristics over the compression function of hash functions. We classify the characteristics into 6 different types, depending on whether the differences in the inputs h_i, m_i and the output h_{i+1} are equal to zero or not.[2] Table 3 presents an overview of the different types and concrete examples from the literature.

To illustrate the connection between this classification and traditional nomenclature [Menezes et al. 1997], we give some examples. A 1-block collision attack on the hash function can be constructed using a type 2 characteristic for the compression function. An n-block collision attack on a hash function can be constructed by using a type 3 characteristic on the first block, a type 6 characteristic on the last block, and type 7 characteristics on the $n-2$ remaining

[2] Note that this gives rise to the 3-bit encoding used for the number in column *Type*.

Table 3: Types of characteristics. 'Y' indicates a non-zero difference, 'N' indicates 'no difference'.

Type	h_i	m_i	h_{i+1}	Examples from literature
2	N	Y	N	MD4 [Wang et al. 2005, Yu et al. 2005]; SHA-0 [Chabaud and Joux 1998, Wang et al. 2005c]
3	N	Y	Y	MD5 [Wang et al. 2005b]; SHA-1 [Biham et al. 2005, Wang et al. 2005a]
4	Y	N	N	MD5 [den Boer and Bosselaers 1993]
5	Y	N	Y	-
6	Y	Y	N	MD5 [Wang et al. 2005b]; SHA-1 [Biham et al. 2005, Wang et al. 2005a]
7	Y	Y	Y	-

blocks. Of course we can't use just any set of such characteristics: the input difference in the chaining variable h of the characteristic in one block needs to match the output difference of the characteristic in the previous block. A 1-block pseudo-collision can be constructed using a type 4 or type 6 characteristic. Similarly, we can use a type 5 or type 7 characteristic in the first block of an n-block pseudo-collision.

In the setting of NMAC/HMAC, many of the characteristics mentioned in Table 3 can not be used. One reason is that their probability is too low (*e. g.* type 6 characteristics for MD5 and SHA-1). Another reason is that for type 3 or type 7 characteristics to be useful additional restrictions on the message difference m_i need to be obeyed. Section 5.2 will cover this issue. Hence the known type 3 characteristics are ruled out as well. The remaining type 2 or type 4 characteristics can be used to draw some conclusions about the security margin offered by a particular hash function when used in HMAC/NMAC. However, we argue that this gives too optimistic conclusions. We use SHA-1 as an example, where a 34-step characteristics is the longest useful characteristics in the literature on collision search. Table 4 gives an overview of new characteristics over the compression function of SHA-1. We developed efficient search algorithms to find them. They are based on methods developed in [Pramstaller et al. 2005], with the improvement that exact probabilities as described in [De Cannière and Rechberger 2006, Mendel et al. 2006] instead of Hamming weights are used to prune and rank them. In Table 4, p_{char} gives the probability of the characteristic with the highest probability. Additionally, the probabilities p_{diff} include the improvements from considering also less-probable characteristics with the same input and output differences, assuming these less-probable characteristics to be independent.

Table 4: Newly presented characteristics over the compression function of SHA-1.

Type	# steps	p_{char}	p_{diff}	details
2	34 (0-33)	2^{-31}	$2^{-30.83}$	Table 2
5	34 (35-69)	2^{-149}	2^{-149}	Table 7
2	37 (19-56)	2^{-66}	$2^{-63.96}$	Table 8
3	50 (0-49)	2^{-72}	$> 2^{-71.15}$	Table 9
2	53 (20-73)	2^{-98}	$2^{-96.11}$	Table 10
7	58 (0-57)	2^{-78}	$> 2^{-77.24}$	Table 11
6	61 (19-79)	2^{-101}	$2^{-99.11}$	Table 12
7	62 (17-78)	2^{-78}	$> 2^{-76.27}$	Table 13

For a characteristic through the compression function of SHA-1 to be of use the probability needs to be significantly higher than 2^{-160}. The reason for including characteristics of the same type but with less steps is that some attacks require characteristics with probability higher than 2^{-80}. The new 50-step type 3 characteristic given in this paper is an extension of the 43-step characteristic used in [Kim et al. 2006]. Note that this is the only characteristic where the characteristics starts at step 0. It is an open problem to find characteristics with high probability spanning more steps including the first steps. Automated approaches to construct characteristics that consider non-linear effects in an efficient way [De Cannière and Rechberger 2006] might serve as important building blocks.

5 New key recovery method for NMAC

Message authentication codes are mappings that have a secret key and a message as input and produce as output a *tag*. If an adversary can produce a valid tag/message pair without knowing the secret key, we speak about a *forgery* attack. Note that if the key itself or equivalent information can be recovered, every message could be forced, this is commonly called *universal forgery*. Let us recall the NMAC construction:

$$NMAC(k_1, k_2, m) = h(k_2, h(k_1, m)). \tag{5}$$

We call the key k_1 and the corresponding h the *inner key* and the *inner hash*. We call k_2 the *outer key* and the corresponding h the *outer hash*. Note that an attacker can act like having the secret key only if *both* the inner key and the outer key (or equivalent information) has been obtained. Hence recovering both the inner and the outer key constitutes a *full key recovery* and allows universal forgery.

Generally speaking, a differential key-recovery attack can work as follows.

off-line preparation phase: Define the characteristic(s).

on-line data collection phase: Obtain the MAC values for pairs of texts with as difference(s) the input difference(s) of the characteristic(s).

off-line data processing phase: When a pair of texts results in a pair of MAC values with as difference the output difference specified by the characteristic(s), assume that this is a right pair. For a right pair, we have information on the intermediate values of the algorithm. This information can be exploited to partially recover the key.

If the characteristic(s) specified a non-zero difference in the key, then the attack is a related-key attack.

5.1 Recovering the inner key

In [Contini and Yin 2006] a related-inner key characteristic is used to recover the inner key of NMAC. After a right pair has been found, their attack proceeds by applying small changes to both messages of the right pair and checking whether the modified pair still results in colliding tags. We will refer to this method as *KR1*. Together with the new characteristics presented in Section 4, we subsequently illustrate that the security margin of HMAC-SHA-1 is less than previously thought.

5.1.1 Example for reduced NMAC-SHA-1

As an example, consider NMAC-SHA-1 where the inner hash is reduced to 61 of its 80 rounds. The best previously published attack applies to NMAC based on SHA-1 reduced to 43 steps. For the recovery of the inner key, we use *KR1* and the new type 6 characteristic given in Table 12. We expect to query a related-key NMAC oracle with 2^{99} message pairs in $\mathcal{M}$ as specified by the characteristic to find a message pair that result in the same MAC. Afterwards, both the effort to recover enough state bits with *KR1* as well as a brute force phase to determine the remaining bits of the inner key k_1 are negligible compared to the online phase. Hence the total complexity is 2^{100} which is significantly less than a 2^{160} black box attack to recover the inner key.

5.2 Recovering the outer key

Once the inner key is recovered, we can attack NMAC to recover the outer key. Different combinations of characteristics over the inner and outer hash function can be used. We list here some possibilities.

1. Type 4 or type 5 over the outer hash combined with the trivial characteristic (input and output differences equal to zero) over the inner hash. This is a related-outer key attack.
2. Type 2, 3, 6 or type 7 over the outer hash combined with type 3, type 5 or type 7 over the inner hash. This is a possibly related-outer key attack with possibly related-inner keys.

In the latter case, the difference in the message input of the characteristic over the outer hash needs to match the (padded) output difference of the characteristic over the inner hash. The right pairs for the inner hash characteristic can be produced off-line; the right pairs for the outer hash characteristic need on-line queries.

5.3 Recovery of the outer key of NMAC-MD5 with *KR1*

We describe here how the inner key recovery attack of [Contini and Yin 2006] can be extended to a full key recovery attack. The same characteristic as for the inner key recovery is used, which has probability 2^{-46}. Note that the conditions in the last 5 steps can be ignored safely, because all resulting differentials can be efficiently enumerated and the output differences can be directly observed. Hence, the sum of their probabilities can be lower bounded by 2^{-41}. Next, *KR1* is used to recover 25 bits of the internal state using $2^{42} + 24 \times 2^{40} \approx 2^{45}$ queries. For each query the first word needs to be controlled, hence requires $25 \times 2^{39+32} \approx 2^{76}$ computations. As described in [den Boer and Bosselaers 1993, Contini and Yin 2006], a 2 bit condition on the IV needs to be met for the attack to work, *i. e.* the most significant bits of three of the four state variables need to be the same. Using this information, the remaining key space can be guessed with $2^{128-2-25} = 2^{101}$ trials. If the actual value of the relevant MSBs is known the number of trials is further reduced to 2^{100} at the cost of another reduction by a factor 0.5 due to conditions on the key. Recovering more bits of the state does not make the attack more efficient since the offline cost to prepare more queries would be higher than the final key guessing. Since the first word is fixed only 3 words (96 degrees of freedom) are left because of the input padding of the outer hash. This is enough to generate the required 2^{44} queries per bit without additional overhead.

5.4 Outline of new key recovery method *KR2*

We propose here an attack strategy *KR2* which can be used to recover first the inner key and subsequently also the outer key of NMAC. In some settings, *KR2* proves to be advantageous, which is shown in an example with step-reduced NMAC-SHA-1.

Suppose we have a suitable characteristic over the outer hash function. Let the set of keys and the set of messages over which the characteristic has improved probability q be denoted by $\mathcal{K}$, respectively $\mathcal{M}$, *i. e.* these are the keys and messages satisfying the easy relations. Furthermore, we assume the probability over the set of messages in $\mathcal{M}$ and the set of keys *not* in $\mathcal{K}$ that a pair of messages produces a collision for the inner hash function to be 2^{-l}. This is the probability for the message pair to produce a collision without being a right pair.

The attack works if $q \gg 2^{-l}$. If after collecting the MAC values for $2q^{-1}$ message pairs in $\mathcal{M}$ with the input difference specified by the characteristic we have observed at least one pair with equal tags, then we conclude that with high probability the key is in $\mathcal{K}$. Otherwise, we conclude that with high probability the key is not in $\mathcal{K}$.

Up to here *KR1* can be reformulated similarly, assuming relations between bits in the state can be efficiently mapped to relations between bits of the key. *KR1* continues to use the same characteristic and submits slightly modified messages in order to deduce more bits of internal state.

The *KR2* method instead uses a set of completely different characteristics and recovers a few key bits with each of them. One key advantage of *KR2* in the outer hash setting is that a factor 2^x for offline computation is saved by not having to fix the first x bits in the input message. It depends on the compression function, whether this outweighs the disadvantage of having less optimal characteristics in the set of characteristics needed for the attack. Another advantage of *KR2* over *KR1* is the increased number of degrees of freedom available: since no message word is fixed it is possible to generate a higher number of distinct queries. This is important in the outer hash setting since here at most l degrees of freedom are given due to the padding of its input.

5.5 Detailed description of *KR2*

The proposed key recovery attack consists of two phases:

1. Online phase: in a chosen-message scenario, the attacker asks for b pairs (m,NMAC(m)) under the same unknown key of length $2l$. Analyzing the results, c linear relations between bits of k are deduced.

2. Offline phase: The rest of the key is guessed in a brute force manner.

The attack is more efficient than brute force, if $2b + 2^{l-c}$ is smaller than 2^l. Subsequently the online phase is described in more detail. Before that, some definitions are needed. Note that the attack applies to HMAC in exactly the same way, expect that instead of the key information, equivalent information is obtained.

Let $\mathcal{K}$ be a set of linear relations between bits in k, and let $p_{\mathcal{K}}$ be the probability that a k picked from a uniform distribution satisfies these linear relations. Likewise, let $\mathcal{M}$ be a set of linear relations in m.

Let q be the probability that there is a collision at the output of the first application of h if m and $m+\alpha$ are input under the assumption that the unknown k satisfies $\mathcal{K}$ and m satisfies $\mathcal{M}$. We write

$$q = \Pr(h(k,m) + h(k,m+\alpha) = 0 \mid k \in \mathcal{K}, \mathtt{m} \in \mathcal{M}). \tag{6}$$

Note that the probability to observe a colliding MAC is higher, namely $q + (1 - q) \cdot 2^{-l}$. Likewise we define q' as the probability for the case that k does not satisfy the relations given by $\mathcal{K}$.

$$q' = \Pr(h(k,m) + h(k,m+\alpha) = 0 \mid k \notin \mathcal{K}, \mathtt{m} \in \mathcal{M}). \tag{7}$$

1. Collect b MAC pairs under the unknown key with chosen messages m and $m + \alpha$ where $m \in \mathcal{M}$.

2. If we observe at least one colliding MAC pair, then $k \in \mathcal{K}$ with probability $1-\epsilon_1$. If we do not observe a colliding MAC pair, then $k \notin \mathcal{K}$ with probability $1 - \epsilon_2$.

3. ϵ_1 can be derived as follows:

$$\begin{aligned}
\Pr(\mathtt{k} \in \mathcal{K} \mid \mathtt{collision}) &= \frac{\Pr(\mathtt{collision} \mid \mathtt{k} \in \mathcal{K}) \Pr(\mathtt{k} \in \mathcal{K})}{\Pr(\mathtt{collision})} \\
&= \frac{(q + (1-q)2^{-l})p_{\mathcal{K}}}{(q + (1-q)2^{-l})p_{\mathcal{K}} + (q' + (1-q')2^{-l})(1-p_{\mathcal{K}})} \\
&= \frac{1}{1 + \frac{(q' + (1-q')2^{-l})(1-p_{\mathcal{K}})}{(q+(1-q)2^{-l})p_{\mathcal{K}}}} \\
&\approx 1 - \frac{(q' + (1-q')2^{-l})(1-p_{\mathcal{K}})}{(q+(1-q)2^{-l})p_{\mathcal{K}}}
\end{aligned}$$

   Note that ϵ_1 is small if q' is small and 2^{-l} is negligible. ϵ_2 can be derived by the approach described in [Kim et al. 2006, Section 6]. ϵ_2 can be made sufficiently small by choosing a high enough b. $b = 2 \cdot q^{-1}$ is enough for practical purposes. For the attack it is important that $q \gg q'$ to ensure a small ϵ_1. Note that given a sufficiently small ϵ_2, ϵ_1 can be estimated to be q'/q. For simplicity we subsequently assume both ϵ_1 and ϵ_2 to be zero.

4. If $k \in \mathcal{K}$, the possible key space is reduced by a factor $p_{\mathcal{K}}^{-1}$. If $k \notin \mathcal{K}$, the possible key space is reduced by a factor $(1 - p_{\mathcal{K}})^{-1}$. For a given $p_{\mathcal{K}}$ the reduction of key entropy is hence

$$p_{\mathcal{K}} \cdot log_2(p_{\mathcal{K}}) + (1 - p_{\mathcal{K}}) \cdot log_2(1 - p_{\mathcal{K}}) \tag{8}$$

bits. Thus the expected reduction in key entropy in this step is at most one bit.

The above described key entropy reduction technique can be applied for any number c of triples $(\alpha_i, \mathcal{K}_i, \mathcal{M}_i)$. To optimize the computational complexity of recovering the full key we choose c such that $2^{l-c} > 2 \cdot \sum_{i=1}^{c} q_i^{-1}$. Note that this assumes the relations between bits in k (*i. e.* $\mathcal{K}$) to be linearly independent.

It remains to be described how to find triples $(\alpha_i, \mathcal{K}_i, \mathcal{M}_i)$ for specific cryptographic hash functions. One way to derive new characteristics for hash functions of the MD4 family like MD5 or SHA-1 is to simply rotate each of the 32-bit words of the inputs of a known characteristic over the same number of bit positions. This follows from two facts. Firstly the linear code describing the message expansion is invariant with respect to word rotation. Secondly, the used characteristic usually requires that there is no carry propagation in the modular addition. This condition has always a probability > 0, although rotation might increase or decrease it. Note that there are special cases of characteristics where this technique does not work for all rotation values.

5.6 Extension of *KR2* to the outer hash setting

Determine a characteristic (h'_{in}, m', h'_{out}) over the outer hash with probability p_2. Probability p_2 for this char needs to be better than 2^{-160+p_1}. We distinguish between two cases:

$m' \neq 0$. Recover the inner key with complexity 2^{p_1}.

Determine a characteristic over the inner hash that produced the output difference that after padding will produce the m' from above. Since the inner key is known at this stage, we can take this into account when constructing the characteristic and during the search for right pairs. Say that the complexity to find a right pair is 2^{p_3}.

We need 2^{p_2} of these near-collisions. Here the offline cost is $2^{p_2+p_3}$. The number of chosen texts is $2^{p_1} + 2^{p_2}$. This allows to recover a certain number of relations between bit of the outer key and potentially can be repeated for some more bits.

$m' = 0$. Note that this implies $h'_{in} \neq 0$: as above but $p_1 = 0$ because we don't need collisions but just single messages. Note that the example for 34-step NMAC-SHA-1 in Section 5.7 is of that type.

5.7 Recovery of the outer key of reduced NMAC-SHA-1 using *KR2*

For HMAC-SHA-1 where the outer hash is reduced to 34 steps, the type 5 characteristics as the one given in Table 7 with probability 2^{-149} can be used[3]. Using *KR2* as proposed in this paper, key recovery is faster than brute force trials. Allowing all possible keys, $p_{\mathcal{K}}$ would be 2^{-20}. We propose a tradeoff between the size of a weak-key class and the possible reduction of key entropy. For a specific class of keys of size 2^{141} all except one of these relations between keybits are met. In this case the probability of the characteristic improves to 2^{-129} and hence 2^{130} queries are enough to reduce the key space by one half.[4]

In order to reduce the key space even more, different characteristics are needed. Many of them can be found that have similar probability but the restriction due to a specific weak-key class usable ones have a lower probability. Every single one of them requires a higher number of online queries and contributes less to a keyspace reduction. Summing up their contributions we arrive at a total number of queries of 2^{155} for a total reduction of the keyspace by a factor of 8.

The remaining cost when using *KR2* for key recovery is 2^{157} guesses. This suggests that a 34-step variant of SHA-1 is already at the borderline between a shortcut attack and a generic brute force attack.

6 Applications and Implications

In the following, we outline how the new characteristics and the key recovery method can be used to analyze popular authentication methods like HMAC or a new proposal for making digital signatures using hash functions safer.

6.1 HMAC

Distinguishing or forgery attacks on NMAC can easily be translated to attacks on HMAC. For key recovery attacks without requiring related keys, instead of the actual key information, equivalent information is obtained. Related-key attacks on NMAC as described in this article can not be translated into attacks on HMAC. Details on attacks exploiting the new method and characteristics developed in this article are given in Table 5 and Table 6. Table entries are either compared to previous results, or are new results for variants with more steps, while success rates of attacks are equalized.

[3] There is a type 5 characteristic for reduced SHA-1 in the literature [Lu et al. 2006], but it contains errors and is hence not used. The one we use is similar but has a slightly lower probability

[4] This is according to Equation 8, since $p_{\mathcal{K}} = 0.5$.

Table 5: Old and new results on attacks on NMAC/HMAC when used with SHA-1. Table entries are either compared to previous results, or are new results for variants with more steps.

	steps	**forgery**	data	truncation	source
HMAC-SHA-1	34 (0-33)	forgery	2^{52}	64	[Kim et al. 2006]
HMAC-SHA-1	34 (0-33)	forgery	2^{34}	64	[Contini and Yin 2006]
HMAC-SHA-1	34 (0-33)	forgery	2^{32}	64	new
HMAC-SHA-1	37 (20-56)	forgery	2^{65}	96	new
	steps	**distinguisher**	data	truncation	source
HMAC-SHA-1	43 $(00-42)$	rectangle d.	$2^{154.9}$	160	[Kim et al. 2006]
HMAC-SHA-1	50 $(00-49)$	rectangle d.	$2^{153.5}$	160	new
HMAC-SHA-1	58 $(00-57)$	related-key rectangle d.	$2^{158.74}$	160	new
HMAC-SHA-1	53 $(20-72)$	differential d.	$2^{97.5}$	128	new
HMAC-SHA-1	61 $(19-79)$	related-key differential d.	2^{100}	128	new
HMAC-SHA-1	62 $(17-78)$	related-key rectangle d.	$2^{157.77}$	128	new

6.1.1 On truncation

Truncating the output of a MAC reduces the amount of bits that need to be guessed in a generic forgery attack and hence reduces the security against this kind of attack. On the other hand certain generic attacks also become harder since an attacker learns less form an output of the used cryptographic building block by received the tag [Preneel and Oorschot 1995]. This tradeoff is widely recommended and commonly done in practice. In both columns labeled 'truncation', we give typical values (multiples of 32 bits) for the size of the truncated output, which still allow to attack the MAC algorithm.

Note that this does not contradict the general statement that truncating helps against certain attacks [Preneel and Oorschot 1995] but is a specific property of the newly devised attacks. In fact, if the output is further truncated than noted, the attack is stopped.

6.1.2 Forgeries

Forgeries for NMAC can be constructed using the same characteristics as for a collision, because a collision for the underlying hash function can be converted trivially into a forgery for NMAC. Naturally, one expects that constructing a forgery for NMAC is more difficult than constructing a collision for the underlying hash function, because now there is a secret key involved. This is correct. However, by sticking to the terminology of differential cryptanalysis when we discussed characteristics, we in fact neglected to take into account the absence of a secret key in a collision attack. Hence, the figures we have given are the ones that are relevant for a forgery attack. For a collision attack (of an unkeyed hash function), they are too pessimistic because if both the message and the chaining variables are known, then state variables can be influenced for many steps (more than 30 in case of SHA-1).

Table 6: Summary of key recovery attacks. They also apply if output is truncated to the number of bits given.

	type	steps	data	offline	truncation	source
NMAC-MD5	inner rel. key	all (0-63)	2^{47}	2^{47}		[Contini and Yin 2006]
HMAC-SHA-1	inner key	34 (0-33)	2^{34}	2^{34}		[Contini and Yin 2006]
NMAC-MD5	full rel. key	all (0-63)	2^{51}	2^{100}		[Fouque et al. 2007]
NMAC-MD5	full rel. key	all (0-63)	2^{45}	2^{100}	64	new
NMAC-SHA-1	full rel. key	34 (0-33)	2^{155}	2^{157}	160	new
HMAC-SHA-1	inner key	34 (0-33)	2^{32}	2^{32}	64	new
NMAC-SHA-1	inner rel. key	61 (19-79)	2^{100}	2^{100}	128	new
HMAC-SHA-1	inner key	53 (20-72)	$2^{99.5}$	$2^{99.5}$	128	new

6.1.3 Distinguishers

When we succeed in constructing a forgery faster than with the black-box attack, we have distinguished NMAC from a pseudo-random function. Hence, a forgery attack implies a distinguishing attack.

Suppose the goal of a distinguisher would be to simply decide if NMAC or HMAC was used to produce a certain set of tags for a number of chosen messages. In such a setting the birthday paradox can be used to construct a distinguisher with only $2^{n/2}$ chosen messages [Kim et al. 2006]. However, if the goal is to distinguish NMAC/HMAC instantiated with a PRF from NMAC/HMAC instantiated with an actual hash function, as listed in Table 5, the new distinguishing attacks can cover more steps than the new forgery attacks. The reason is that no generic distinguisher is known that would require less than 2^n messages [Kim et al. 2006].

Also, a distinguishing attack doesn't need to be based on a collision. Also near-collisions can be used to distinguish NMAC from a pseudo-random function as shown by Kim *et al.* who build rectangle distinguishers. As shown in Table 5 we also improve on this attack by simply extending the used characteristic for 7 more steps and apply some of the improvements mentioned earlier in the article.

6.1.4 Key recovery

In Table 6 we summarize the new attacks that recover the full key of NMAC when observing a number of text/MAC pairs. Some attacks require related keys. We also add attacks that recover only the inner key, while noting that this allows forgery attacks but does not give an attacker the same possibilities as having the full key.

6.2 Randomized hashing

The RMX mode of operation is proposed as a means to provide a safety net in applications relying on hash functions, by reducing the impact of collisions

[Halevi and Krawczyk 2006]. Basically the proposal is to prepend a random string of length equal to the block length of the compression function to the message that is supposed to be hashed. Additionally, every message block m_i is XORed with the same random message block.

$$RMX(r, m) = H(r||m_1 \oplus r|| \dots ||m_n \oplus r) \tag{9}$$

We briefly discuss the applicability of our results to this mode. Put in a simple way, a hash function used in RMX mode doesn't need to be collision resistant. Second preimage resistance, or more precisely *e-SPR* resistance, is sufficient [Halevi and Krawczyk 2006]. A compression function h is defined to be e-SPR resistant if no method exists that allows to output a y such that $RMX(r, x) = RMX(r, y)$ if r and x are not under control of the attacker.

The natural question arises, namely to which extend are popular hash function e-SPR resistant? We explain here how our characteristics can be used in a preimage attack. For a second preimage attack on 53-step HMAC-SHA-1, we can reuse the characteristic presented in Appendix A. In a differential second preimage attack, only one pair can be tried for each characteristic. Furthermore, there is no distinction between easy relations and others. Hence the probability of the characteristic is reduced to $2^{-151.5}$, and this is also the probability of success of the attack. Note that considering also less probable characteristics as described in Section 3.5 would allow some improvements of this probability.

Note that if the first preimage consists of t message blocks, we can try our characteristic once in each of the t blocks, and hence multiply the probability of success by t. Our results imply that SHA-1 reduced to 53 steps is not as e-SPR resistant as an ideal compression function used in the proof of security of [Halevi and Krawczyk 2006].

7 Conclusions and Future Work

We presented a thorough security evaluation of the heavily used authentication method HMAC when used with MD5 and SHA-1. Even though recent results on the collision resistance of the employed hash function triggered renewed interest in the security offered by HMAC when used with the affected hash functions, our results are more general. Using our newly developed key recovery method it turns out that in addition to collision attacks, also other non-random properties of the employed hash function can be used.

The results are the first full key recovery attack for NMAC-MD5 and a decreased security margin offered by HMAC-SHA-1. Most of the attacks work even if the output of the MAC is truncated, which is commonly done in practice. It is an open problem if automated methods that efficiently include non-linear effects while searching for useful characteristics can be used to improve on the attacks presented in this article.

Despite the progress being made, message authentication algorithms like HMAC are less susceptible to problems in the underlying hash function than the standalone hash function. However, it seems prudent to evaluate other hash functions like RIPEMD-160 or members of the SHA-2 family as well as new hash function proposals against the framework of undesired properties as shown in this paper.

Acknowledgements

We would like to thank Christophe De Cannière, Florian Mendel, Lars R. Knudsen, Hugo Krawczyk, and Norbert Pramstaller for helpful discussions.

The work described in this paper has been supported in part by the European Commission through the IST Programme under contract IST2002507 932 ECRYPT[5] and in part by the Austrian Science Fund (FWF), project P18138.

References

[Bellare et al. 1996] Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In Koblitz, N., editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, 1996, Proceedings*, volume 1109 of *LNCS*, pages 1–15. Springer, 1996.

[Bellare 2006] Bellare, M.: New Proofs for NMAC/HMAC. In Dwork, C., editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *LNCS*, pages 602–619. Springer, 2006.

[Biham et al. 2005] Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby W.: Collisions of SHA-0 and Reduced SHA-1. In Cramer, R., editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 36–57. Springer, 2005.

[Biham and Shamir 1991] Biham, E., Shamir, A.,: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology, 4(1):3–72, 1991.

[Chabaud and Joux 1998] Chabaud, F,: Joux, A.: Differential Collisions in SHA-0. In Krawczyk, H., editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *LNCS*, pages 56–71. Springer, 1998.

[Contini and Yin 2006] Contini, S., Yin, Y.L.: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In Lai X., Chen, K., editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *LNCS*, pages 37–53. Springer, 2006.

[Damgård 1989] Damgård, I.: A Design Principle for Hash Functions. In Brassard, G., editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *LNCS*, pages 416–427. Springer, 1989.

[5] The information in this paper is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

[den Boer and Bosselaers 1993] den Boer, B., Bosselaers, A.: Collisions for the Compressin Function of MD5. In Helleseth, T., editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *LNCS*, pages 293–304, 1993.

[De Cannière et al. 2007] De Cannière, C., Mendel, F., Rechberger, C.: Collisons for 70-step SHA-1: On the Full Cost of Collision Search. In Adams, C.M., Miri, A., Wiener, M.J., editors, *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, volume 4876 of *LNCS*, pages 56–73. Springer, 2007.

[De Cannière and Rechberger 2006] De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In Lai X., Chen, K., editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.

[Fouque et al. 2007] Fouque, P., Leurent, G., Nguyen, P.Q.: Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In Menezes, A., editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *LNCS*, pages 13–30. Springer, 2007.

[Halevi and Krawczyk 2006] Halevi, S., Krawczyk, H.: Strengthening Digital Signatures via Randomized Hashing. In Dwork, C., editor, *Advances in Cryptology - CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 2006, Proceedings*, volume 4117 of *LNCS*, pages 41–59. Springer, 2006.

[Kim et al. 2006] 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 (Extended Abstract). In De Prisco, R., Yung, M., editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2006.

[Lu et al. 2006] Lu, J., Kim, J., Keller, N., Dunkelman, O.: Differential and Rectangle Attacks on Reduced-Round SHACAL-1. In Barua, R., Lange, T., editors, *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*, volume 4329 of *LNCS*, pages 17–31. Springer, 2006.

[Mendel et al. 2006] Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: The Impact of Carries on the Complexity of Collision Attacks on SHA-1. In Robshaw, M., editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Pre-Proceedings*, 2006.

[Menezes et al. 1997] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: *Handbook of Applied Cryptography*. CRC Press, 1997. Available online at `http://www.cacr.math.uwaterloo.ca/hac/`.

[Merkle 1989] Merkle, R.C.: One Way Hash Functions and DES. In Brassard, G., editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

[NIST 2002] National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002. Available online at `http://www.itl.nist.gov/fipspubs/`.

[NIST 2006] National Institute of Standards and Technology. NIST's Policy on Hash Functions, 2006. Available online at `http://www.csrc.nist.gov/pki/HashWorkshop/NIST%20Statement/NIST_Policy_on_HashFunctions.htm`.

[Pramstaller et al. 2005] Pramstaller, N., Rechberger, C., Rijmen, V.: Exploiting Coding Theory for Collision Attacks on SHA-1. In Smart, N.P., editor, *Cryptography* *and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *LNCS*, pages 78–95. Springer, 2005.

[Preneel and Oorschot 1995] Preneel, B., van Oorschot, P.C.: MDx-MAC and Building Fast MACs from Hash Functions. In Coppersmith, D., editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *LNCS*, pages 1–14. Springer, 1995.

[Raihi et al. 2005] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., Ranen, O.: HOTP: An HMAC-based One Time Password Algorithm. Informational RFC 4226, December 2005.

[Rechberger and Rijmen 2007] Rechberger, C.: Rijmen, V.: On Authentication with HMAC and Non-Random Properties. In Dietrich, S., Dhamija, R., editors, *Proceedings of Financial Cryptography 2007 and Usable Security 2007, Trinidad and Tobago, February 12-15, 2007*, volume 4886 of *LNCS*, pages 119–133. Springer, 2007.

[Wang 2005] Wang, X.: What's the Potential Danger Behind the collisions of Hash Functions, 2005. ECRYPT Conference on Hash Functions, Krakow, available via `http://www.ecrypt.eu.org/stvl/hfw/`.

[Wang et al. 2005] Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Cramer, R., editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.

[Wang et al. 2005a] Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In Shoup, V., editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.

[Wang et al. 2005b] Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In Cramer, R., editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

[Wang et al. 2005c] Wang, X., Yu, H., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In Shoup, V., editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 1–16. Springer, 2005.

[Yu et al. 2005] Yu, H., Wang, G., Zhang, G., Wang, X.: The Second-Preimage Attack on MD4. In Desmedt, Y., Wang, H., Mu, Y., and Li, Y, editors, *Cryptology and Network Security, 4th International Conference, CANS 2005, Xiamen, China, December 14-16, 2005, Proceedings*, volume 3810 of *LNCS*, pages 1–12. Springer, 2005.

A Characteristics

Table 7: Type 5 characteristic with probability 2^{-149} used for the 34-step (35-69) attack.

i	∇A_i	∇W_i	$P_c(i)$
-4	x------x-x-x----x--x---x------x-		
-3	--x----x---x----------x----x-x--		
-2	x-x---x--x------x------x---x-x--		
-1	-------x---x---------x-x--------		
0	--x---xx---x-----------------x--	--------------------------------	18
1	--n--------u-----------n-----n--	--------------------------------	12
2	-----------------------u--------	--------------------------------	8
3	--------------------------------	--------------------------------	6
4	-------------n------------------	--------------------------------	6
5	---------u----------------------	--------------------------------	3
6	-------------------------------n	--------------------------------	3
7	--------------------------u-----	--------------------------------	4
8	---------------------u----------	--------------------------------	4
9	----------------n---------------	--------------------------------	3
10	--------------------------------	--------------------------------	4
11	-n------------------------------	--------------------------------	2
12	--------------------------------	--------------------------------	3
13	-----------------------u--------	--------------------------------	1
14	--------------------------------	--------------------------------	2
15	--------------------------------	--------------------------------	2
16	---n----------------------------	--------------------------------	2
17	------------------------------n-	--------------------------------	1
18	--------------------------------	--------------------------------	2
19	--------------------------------	--------------------------------	2
20	--------------------------------	--------------------------------	2
21	-----n--------------------------	--------------------------------	0
22	--------------------------------	--------------------------------	1
23	--------------------------------	--------------------------------	1
24	--------------------------------	--------------------------------	1
25	--------------------------------	--------------------------------	1
26	-------n------------------------	--------------------------------	1
27	--n-----------------------------	--------------------------------	3
28	-------n---------------------n--	--------------------------------	4
29	---------n--------------n-------	--------------------------------	8
30	-------n-n---------n---------u--	--------------------------------	7
31	--n------n----n----------------n	--------------------------------	12
32	---n---u-n-n-------u-----------u	--------------------------------	10
33	----n-n--n-----------n-------n-u	--------------------------------	10
34	-------x--n--------x-x--n------n		

Table 8: Type 2 characteristic with probability 2^{-66} used for the 37-step (19-56) attack.

i	∇A_i	∇W_i	$P_c(i)$
-4	`--------------------------------`		
-3	`--------------------------------`		
-2	`--------------------------------`		
-1	`--------------------------------`		
0	`--------------------------------`	`------------------------------un`	1
1	`-------------------------------u`	`-------------------------nu-----`	0
2	`--------------------------------`	`n-------------------------------`	3
3	`u------------------------------n`	`uu-----------------------unn----`	1
4	`--------------------------------`	`-u-----------------------------n`	3
5	`------------------------------n-`	`--n----------------------u------`	2
6	`--------------------------------`	`-nu---------------------------n-`	4
7	`n-------------------------------`	`-nn------------------------u--u-`	1
8	`------------------------------u-`	`-------------------------n-----u`	1
9	`-------------------------------u`	`n1u-----------------------n----u`	4
10	`------------------------------nu`	`u-u----------------------un----u`	2
11	`--------------------------------`	`nuu----------------------------u`	4
12	`------------------------------n-`	`-0-----------------------u----n-`	1
13	`------------------------------n-`	`n------------------------u----n-`	2
14	`--------------------------------`	`-n------------------------0----n`	2
15	`-------------------------------u`	`-------------------------nu-----`	0
16	`--------------------------------`	`-1------------------------------`	2
17	`-------------------------------n`	`-n-----------------------un-----`	1
18	`--------------------------------`	`un------------------------------`	3
19	`-------------------------------u`	`-------------------------nu---n-`	2
20	`------------------------------n-`	`un-----------------------u----n-`	3
21	`-------------------------------n`	`--------------------------u-----`	3
22	`------------------------------n-`	`-n-----------------------u------`	4
23	`-------------------------------n`	`--------------------------u---u-`	3
24	`--------------------------------`	`-n-----------------------------u`	3
25	`--------------------------------`	`n-------------------------------`	2
26	`--------------------------------`	`nu----------------------------u-`	2
27	`------------------------------u-`	`-u-----------------------n------`	0
28	`--------------------------------`	`--------------------------1---n-`	1
29	`--------------------------------`	`n-------------------------------`	1
30	`--------------------------------`	`n-----------------------------n-`	2
31	`------------------------------n-`	`u0-----------------------u------`	0
32	`--------------------------------`	`------------------------------u-`	1
33	`--------------------------------`	`n----------------------0--------`	1
34	`--------------------------------`	`u------------------------------1`	1
35	`--------------------------------`	`u-------------------------------`	0
36	`--------------------------------`	`--------------------------------`	0
37	`--------------------------------`		

Table 9: Type 3 characteristic with probability 2^{-72} used for the 50-step (0-49) attack.

i	∇A_i	∇W_i	$P_c(i)$
-4	--------------------------------		
-3	--------------------------------		
-2	--------------------------------		
-1	----------------------------1---		
0	----------------------------0---	------------------------------u-	2
1	----------------------------0-u-	-------------------------n------	2
2	1---------------------------1---	--------------------------------	2
3	0-----------------------------n-	u------------------------u------	1
4	1-------------------------------	n-------------------------------	3
5	0----------------------------0n-	-------------------------u------	2
6	1----------------------------1--	n------------------------------n	3
7	0------------------------------n	-------------------------un-----	2
8	-1--------------------------0---	n-----------------------------un	2
9	-0--------------------------1---	-n----------------------------n-	1
10	------------------------------n-	un-----------------------u------	1
11	1-------------------------------	un----------------------------u-	1
12	0-------------------------------	u-------------------------------	1
13	----------------------------1---	n-------------------------------	1
14	----------------------------0---	u-----------------------------n-	1
15	------------------------------n-	-------------------------u------	1
16	1-------------------------------	------------------------------n-	1
17	0-------------------------------	u-------------------------------	0
18	--------------------------------	n-------------------------------	0
19	--------------------------------	n-------------------------------	0
20	--------------------------------	------------------------------n0	1
21	------------------------------n-	-------------------------u-----0	0
22	--------------------------------	-------------------------------1	2
23	------------------------------n-	n------------------------u------	0
24	--------------------------------	u-----------------------------u-	1
25	--------------------------------	--------------------------------	0
26	--------------------------------	n------------------------------0	0
27	--------------------------------	u------------------------------0	0
28	--------------------------------	--------------------------------	0
29	--------------------------------	--------------------------------	0
30	--------------------------------	--------------------------------	0
31	--------------------------------	--------------------------------	0
32	--------------------------------	-----------------------------10-	0
33	--------------------------------	--------------------------------	0
34	--------------------------------	1----------------------------n--	1
35	-----------------------------n--	------------------------u-------	0
36	--------------------------------	0----------------------------n--	1
37	--------------------------------	----------------------------u--u	2
38	----------------------------u---	-----------------------n-----u-n	2
39	-----------------------------u--	------------------------n---n--u	1
40	--------------------------------	---------------------------n-nu-	3
41	---------------------------n----	----------------------u-----n-nn	3
42	----------------------------n---	-----------------------u---uu-nn	3
43	----------------------------u---	-----------------------n--n-uu-n	3
44	--------------------------n-----	---------------------u-----uunu-	4
45	---------------------------u----	----------------------n---u--u--	1
46	--------------------------------	-------------------------n-u----	5
47	-------------------------n--u---	--------------------u--n--u-unu-	3
48	--------------------------u-----	---------------------n---uu-nu--	5
49	--------------------------u-n---	---------------------n-uu-nu-nn-	4
50	------------------------u-------		

Table 10: Type 2 characteristic with probability 2^{-98} used for the 53-step (20-73) attack.

i	∇A_i	∇W_i	$P_c(i)$
-4	--------------------------------		
-3	--------------------------------		
-2	--------------------------------		
-1	--------------------------------		
0	--------------------------------	-0------------------------------	0
1	--------------------------------	-0------------------------------	0
2	--------------------------------	1-------------------------------	0
3	--------------------------------	1n0-----------------------------	1
4	-n------------------------------	0-0-------------------------u---	0
5	--------------------------------	1n1---------------------------n-	2
6	------------------------------n-	u--u---------------------u------	2
7	u-------------------------------	0n0u-----------------------n---n	4
8	-n-----------------------------u	1-0u---------------------nu-u---	0
9	--------------------------------	nnn---------------------------u-	4
10	-------------------------------u	nnun----------------------n---u-	5
11	u-----------------------------u-	1nnn---------------------n-n---n	4
12	u-------------------------------	101u-----------------------n----	3
13	------------------------------n-	0nu----------------------u-----u	3
14	-------------------------------u	nn------------------------n---u-	1
15	--------------------------------	n0----------------------------nu	4
16	u-----------------------------n-	nun----------------------u-n--n-	2
17	------------------------------n-	nn-----------------------u------	4
18	n-----------------------------u-	unn----------------------n-u--n1	2
19	--------------------------------	11n---------------------------nu	4
20	n------------------------------u	u1------------------------nu---0	1
21	--------------------------------	00u----------------------------n	3
22	u-------------------------------	uu-------------------------n--u1	3
23	------------------------------u-	1un----------------------n-----u	5
24	n------------------------------u	-n0-----------------------nu--n1	2
25	--------------------------------	10n---------------------------nn	3
26	------------------------------n-	un-----------------------u----u1	4
27	------------------------------u-	uun----------------------n----u-	3
28	--------------------------------	unu---------------------------n1	2
29	--------------------------------	010------------------------1--u1	1
30	------------------------------u-	11-----------------------n---1-0	1
31	--------------------------------	u01----------------------------n	2
32	-------------------------------u	n00----------------------nu--1--	1
33	--------------------------------	n11----------------------------u	3
34	------------------------------u-	1u1----------------------n----n0	2
35	------------------------------n-	nu1----------------------u----00	3
36	------------------------------u-	1n0----------------------n----u1	2
37	--------------------------------	10----------------------------00	2
38	------------------------------u-	u0-----------------------n----0-	0
39	--------------------------------	11---------------------------0nn	3
40	-------------------------------n	0-1-----------------------u-----	0
41	--------------------------------	n-----------------------------1u	1
42	--------------------------------	nn--------------------------01n0	2
43	------------------------------n-	-n-----------------------u----11	1
44	--------------------------------	-u----------------------------u1	1
45	--------------------------------	u-----------------------------u0	1
46	------------------------------u-	u-----------------------1n----0-	0
47	--------------------------------	u1---------------------------1n-	1
48	--------------------------------	n-----------------------------10	0
49	--------------------------------	n------------------------------1	0
50	--------------------------------	n---------------------------01-1	0
51	--------------------------------	-----------------------------01-	0
52	--------------------------------	--------------------------------	0
53	--------------------------------		

Table 11: Type 7 characteristic with probability 2^{-78} used for the 58-step (00-57) attack.

i	∇A_i	∇W_i	$P_c(i)$
-4	n-------------------------------		
-3	---------------------0-------n--		
-2	n--------------------1----------		
-1	--1--------------------u----0--1		
0	n-0----------------------1--n---	un------------------------un--n-	3
1	--1----------------------1----n-	1-u----------------------------u	6
2	1-0-------------------------0-1-	un----------------------------u-	2
3	1---------------------------1---	nun----------------------n----u-	1
4	------------------------------u-	-un----------------------n----u-	1
5	0-------------------------------	u-----------------------------n-	1
6	1-------------------------------	--------------------------------	0
7	--------------------------------	--------------------------------	1
8	----------------------------0---	u-------------------------------	1
9	----------------------------1---	------------------------------n-	2
10	----------------------------1-n-	-------------------------u------	2
11	1---------------------------0---	--------------------------------	2
12	0-----------------------------n-	n------------------------u------	1
13	1-------------------------------	n-------------------------------	3
14	0----------------------------1u-	-------------------------n------	2
15	1----------------------------0--	u------------------------------n	3
16	0------------------------------n	-------------------------un----1	1
17	-1------------------------------	u-----------------------------nu	1
18	-0------------------------------	-n----------------------------u-	1
19	------------------------------u-	nn-----------------------n------	0
20	--------------------------------	nn----------------------------n-	1
21	--------------------------------	n-------------------------------	0
22	--------------------------------	n-------------------------------	0
23	--------------------------------	n-----------------------------n-	1
24	------------------------------n-	-------------------------u------	0
25	--------------------------------	------------------------------u-	1
26	--------------------------------	u-------------------------------	0
27	--------------------------------	u-------------------------------	0
28	--------------------------------	n------------------------------1	0
29	--------------------------------	------------------------------u0	1
30	------------------------------u-	-------------------------n-----0	0
31	--------------------------------	-------------------------------0	2
32	------------------------------u-	n------------------------n------	0
33	--------------------------------	n-----------------------------n-	1
34	--------------------------------	--------------------------------	0
35	--------------------------------	u------------------------------1	0
36	--------------------------------	u------------------------------0	0
37	--------------------------------	--------------------------------	0
38	--------------------------------	--------------------------------	0
39	--------------------------------	--------------------------------	0
40	--------------------------------	--------------------------------	0
41	--------------------------------	-----------------------------10-	0
42	--------------------------------	--------------------------------	0
43	--------------------------------	0----------------------------u--	1
44	-----------------------------u--	------------------------n-------	0
45	--------------------------------	0----------------------------n--	1
46	--------------------------------	----------------------------n--n	2
47	----------------------------n---	-----------------------u-----u-n	2
48	-----------------------------u--	------------------------n---u--n	1
49	--------------------------------	---------------------------n-nu-	3
50	---------------------------n----	----------------------u-----u-un	3
51	----------------------------u---	-----------------------n---un-un	3
52	----------------------------n---	-----------------------u--u-nu-n	3
53	--------------------------u-----	---------------------n-----nnuu-	4
54	---------------------------n----	----------------------u---n--u--	1
55	--------------------------------	-------------------------u-n----	5
56	-------------------------u--n---	--------------------n--u--n-nun-	3
57	--------------------------n-----	---------------------u---un-un--	5
58	--------------------------u-u---		

Table 12: Type 6 characteristic with probability 2^{-101} used for the 61-step (19-79) attack.

i	∇A_i	∇W_i	$P_c(i)$
-4	--------------------------------		
-3	------------------------------u-		
-2	--------------------------------		
-1	0-------------------------------		
0	--------------------------------	u-1-----------------------------	0
1	--------------------------------	n1------------------------------	0
2	--------------------------------	-1------------------------------	0
3	--------------------------------	10------------------------------	0
4	--------------------------------	1-------------------------------	0
5	--------------------------------	1u0-----------------------------	1
6	-u------------------------------	0-0-------------------------n---	0
7	--------------------------------	0n1---------------------------n-	2
8	------------------------------n-	u--n---------------------u------	2
9	n-------------------------------	0n0u-----------------------u---u	4
10	-n-----------------------------n	1-1n---------------------un-u--0	0
11	--------------------------------	nuu---------------------------n0	4
12	-------------------------------n	uunu----------------------u---u-	5
13	n-----------------------------u-	0nuu---------------------n-u---u	4
14	u-------------------------------	101u-----------------------n----	3
15	------------------------------n-	1nu----------------------u-----n	3
16	-------------------------------n	uu------------------------u---n-	1
17	--------------------------------	u0----------------------------un	4
18	u-----------------------------u-	unn----------------------n-n--u-	2
19	------------------------------u-	un-----------------------n-----1	4
20	n-----------------------------n-	uun----------------------u-u--u1	2
21	--------------------------------	10n---------------------------un	4
22	u------------------------------u	n1------------------------nn---1	1
23	--------------------------------	10u----------------------------n	3
24	n-------------------------------	un-------------------------u--n0	3
25	------------------------------n-	0nn----------------------u----0u	5
26	u------------------------------u	-n0-----------------------nn--u0	2
27	--------------------------------	10u---------------------------nn	3
28	------------------------------n-	nu-----------------------u----u1	4
29	------------------------------u-	nnn----------------------n----u-	3
30	--------------------------------	nnn---------------------------n1	2
31	--------------------------------	101------------------------1--u1	1
32	------------------------------u-	11-----------------------n---0-1	1
33	--------------------------------	n01---------------------------1n	2
34	-------------------------------n	u11----------------------un--0--	1
35	--------------------------------	u01---------------------------0n	3
36	------------------------------n-	1n0----------------------u----u1	2
37	------------------------------u-	un1----------------------n----11	3
38	------------------------------n-	1n1----------------------u----n0	2
39	--------------------------------	00----------------------------11	2
40	------------------------------n-	u0-----------------------u----0-	0
41	--------------------------------	01---------------------------1un	2
42	-------------------------------n	0-1-----------------------u-----	0
43	--------------------------------	n----------------------------11u	1
44	--------------------------------	nn--------------------------01u1	2
45	------------------------------u-	-u-----------------------n----10	1
46	--------------------------------	-u----------------------------u0	1
47	--------------------------------	n-1--------------------------0u1	1
48	------------------------------u-	u-----------------------0n--1-1-	0
49	--------------------------------	n1--------------------------01n-	1
50	--------------------------------	u1----------------------------00	0
51	--------------------------------	u----------------------------1-1	0
52	--------------------------------	u-0-------------------------10-0	0
53	--------------------------------	0---------------------------110-	0
54	--------------------------------	--------------------------------	0
55	--------------------------------	------------------------------n1	1
56	------------------------------n-	-------------------------u--1-10	0
57	--------------------------------	------------------------------1-	2
58	------------------------------n-	u------------------------u------	0
59	--------------------------------	u---------------------------0-n-	1
60	--------------------------------	----------------------------1---	0
61	--------------------------------		

Table 13: Type 7 characteristic with probability 2^{-78} used for the 62-step (17-78) attack.

i	∇A_i	∇W_i	$P_c(i)$
-4	--------------------------------		
-3	----------------------------0---		
-2	u---------------------------1-0-		
-1	--1-------------------------0-n-		
0	n-0---------------------------u-	nuu----------------------n-u--u-	1
1	--1-----------------------------	n-n-----------------------------	1
2	------------------------------n-	u------------------------u------	0
3	--------------------------------	--n----------------------------u	3
4	-------------------------------n	--u----------------------un-----	0
5	--------------------------------	u------------------------------u	2
6	------------------------------u-	-u-----------------------n----n-	2
7	------------------------------n-	uu-----------------------u----nn	3
8	-------------------------------n	-n------------------------u---n-	1
9	--------------------------------	-0----------------------------uu	2
10	------------------------------u-	-n-----------------------n----u-	2
11	------------------------------u-	nn-----------------------n----nu	3
12	-------------------------------u	uu------------------------n---u-	1
13	--------------------------------	-------------------------------n	1
14	--------------------------------	-u----------------------------n-	2
15	------------------------------n-	nu-----------------------u----nu	2
16	-------------------------------n	-n-----------------------un---u0	1
17	--------------------------------	n------------------------------n	2
18	------------------------------n-	-n----------------------1u----u0	2
19	------------------------------u-	-u-----------------------n----u-	2
20	--------------------------------	-n----------------------------n0	1
21	--------------------------------	-1----------------------------u-	1
22	------------------------------u-	1------------------------n---1-1	0
23	--------------------------------	n0---------------------------1n-	1
24	--------------------------------	u----------------------------1--	1
25	--------------------------------	u1----------------------------u-	2
26	------------------------------u-	u------------------------n---1--	0
27	--------------------------------	1-------------------------------	2
28	------------------------------u-	u------------------------n---0--	1
29	--------------------------------	n-------------------------------	3
30	------------------------------u-	-------------------------n------	1
31	--------------------------------	u------------------------------1	3
32	------------------------------u-	-----------------------0-n------	1
33	--------------------------------	n1----------------------------n-	2
34	--------------------------------	-----------------------0--------	1
35	--------------------------------	u-------------------------------	1
36	--------------------------------	n-------------------------------	0
37	--------------------------------	--------------------------------	0
38	--------------------------------	-----------------------0--------	0
39	--------------------------------	----------------------------0-10	0
40	--------------------------------	--------------------------------	0
41	--------------------------------	----------------------------1---	0
42	--------------------------------	--------------------------------	0
43	--------------------------------	--------------------------------	0
44	--------------------------------	--------------------------------	0
45	--------------------------------	--------------------------------	0
46	--------------------------------	--------------------------------	0
47	--------------------------------	----------------------------0u--	1
48	-----------------------------u--	------------------------n-------	0
49	--------------------------------	-----------------------------n--	1
50	--------------------------------	----------------------------n--n	2
51	----------------------------n---	-----------------------u-------u	1
52	--------------------------------	----------------------------n--n	1
53	--------------------------------	---------------------------n--u-	2
54	---------------------------n----	----------------------u-------n-	1
55	--------------------------------	---------------------------uu-u-	2
56	----------------------------u---	-----------------------n--n--n--	2
57	--------------------------n-----	---------------------u------nn--	2
58	--------------------------------	--------------------------u--un-	2
59	--------------------------------	-------------------------u--n-n-	3
60	-------------------------u------	--------------------n-------n-n-	1
61	--------------------------------	-------------------------uu-----	3
62	---------------------------uu---		

Table 14: Detailed probabilities of local collisions, assuming no easy relations to be set.

i	31	30	29	28 . . . 27	26	25 . . . 3	2	1	0
01. . . 16	7.00	8.81	8.92	8.92	8.00	8.86	8.92	6.00	8.75
17	6.00	7.87	7.95	7.95	7.00	7.90	7.95	5.00	7.75
18	5.00	6.91	6.97	6.97	6.00	6.93	6.97	4.00	6.75
19. . . 36	4.00	5.91	5.98	5.98	5.00	5.96	5.98	3.00	5.83
37	5.00	6.87	6.96	6.97	6.00	6.93	6.97	4.00	6.83
38	6.00	7.81	7.95	7.95	7.00	7.90	7.95	5.00	7.83
39. . . 56	7.00	8.81	8.92	8.92	8.00	8.86	8.92	6.00	8.75
57	6.00	7.87	7.95	7.95	7.00	7.90	7.95	5.00	7.75
58	5.00	6.91	6.97	6.97	6.00	6.93	6.97	4.00	6.75
59. . . 75	4.00	5.91	5.98	5.98	5.00	5.96	5.98	3.00	5.83

Table 15: Detailed probabilities of local collisions, assuming easy relations can be set.

i	31	30	29	28 . . . 27	26	25 . . . 3	2	1	0
01	4.00	2.96	2.96	2.96	3.00	2.97	2.96	3.00	2.97
02	4.00	3.91	3.93	3.93	4.00	3.93	3.93	4.00	3.93
03. . . 16	4.00	4.83	4.86	4.86	5.00	4.87	4.86	5.00	4.87
17	4.00	4.83	4.86	4.86	5.00	4.87	4.86	4.00	4.75
18	4.00	4.83	4.86	4.86	5.00	4.87	4.86	3.00	4.54
19. . . 36	3.00	3.83	3.90	3.91	4.00	3.91	3.91	2.00	3.68
37	3.00	3.83	3.90	3.91	4.00	3.91	3.91	3.00	3.83
38	3.00	3.83	3.90	3.91	4.00	3.91	3.91	4.00	3.91
39. . . 56	4.00	3.91	3.91	3.91	4.00	3.91	3.91	4.00	3.91
57	4.00	3.91	3.91	3.91	4.00	3.91	3.91	3.00	3.83
58	4.00	3.91	3.91	3.91	4.00	3.91	3.91	2.00	3.68
59. . . 75	3.00	3.83	3.90	3.91	4.00	3.91	3.91	2.00	3.68