On Authentication with HMAC and Non-Random Properties*

Christian Rechberger and Vincent Rijmen

Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A–8010 Graz, Austria
Christian.Rechberger@iaik.tugraz.at
www.iaik.tugraz.at/research/krypto/

Abstract. 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 61 out of 80). This information about the new, reduced security margin serves as an input to the selection of algorithms for authentication purposes.

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 [1] 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 collision attacks on MD5 [21] and reduced versions of SHA-1 [20], the impact of new collision attacks on the security of MAC constructions needs to be considered. This issue was already briefly mentioned in [23] and the impact on early hash based MAC constructions like the prefix-, suffix-, or envelope-method was informally discussed in [18]. Of particular interest is the impact on HMAC, since *e. g.* NIST supports HMAC-SHA-1 even after 2010 [14], whereas support for SHA-1 as a hash function will be dropped. In addition HMAC-SHA-1 is continued to be used in new designs and applications, *e. g.* in [13].

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. On the other hand, earlier work [6, 9] 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.

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

* A shortened version of this paper appears in [17].

1. Previous work on the security of NMAC and HMAC [6, 9] 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 3. 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 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 4. 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 5. Conclusions and open problems are given in Section 6.

2 Characteristics and probabilities

2.1 Terminology from differential cryptanalysis

Differential cryptanalysis was originally invented to attack DES and other block ciphers [3]. 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.

2.2 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.* However, when

the message is under full control of the attacker, optimizations are possible. It was already observed in the early analysis of SHA [5] 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. Characteristics for SHA-1 are built up of disturbances and their corresponding local collisions. 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 11 and 12. 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 [11]. In the attacks presented in this paper, these effects on the probability of the characteristics are taken into consideration.

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

For example, we can increase the probability of the 34-step SHA-1 characteristic as presented in [9] from 2^{-52} to 2^{-31}. See Table 6 for details.

2.3 Multiple characteristics in one 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 [11]. Tables 11 and 12 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 4. 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.

3 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. Table 2 presents an overview of the different types and concrete examples from the literature.

Table 2. 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 [19, 23]; SHA-0, [5, 22]
3	N	Y	Y	MD5 [21]; SHA-1 [2, 20]
4	Y	N	N	MD5 [7]
5	Y	N	Y	reduced SHA-1 [10]
6	Y	Y	N	MD5 [21]; SHA-1 [2, 20]
7	Y	Y	Y	

To illustrate the connection between this classification and traditional nomenclature [12], we give some examples. A 1-block collision attack on the hash function can be constructed by 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 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 2 can not be used. On 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 4.2 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 functions 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 3 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 [15], with the improvement that exact probabilities as described in [4, 11] instead of Hamming weights are used to prune and rank them. In Table 3, 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.

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

Table 3. Newly presented characteristics over the compression function of SHA-1.

Type	# steps	p_{char}	p_{diff}	details
2	34	2^{-31}	$2^{-30.83}$	Table 6
2	37	2^{-66}	$2^{-63.96}$	Table 7
3	50	2^{-72}	2^{-72}	Table 8
2	53	2^{-98}	$2^{-96.11}$	Table 9
6	61	2^{-101}	$2^{-99.11}$	Table 10

given in this paper is an extension of the 43-step characteristic used in [9]. Note also 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 [4] might serve as important building blocks.

4 New key recovery method for NMAC

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*. 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*.

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.

4.1 Recovering the inner key

In [6] 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 3, we subsequently illustrate that the security margin of HMAC-SHA-1 is less than previously thought.

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 10. 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.

4.2 Recovering the outer key

Once the inner key has been recovered, also the outer key can be attacked. 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.

Recovery of the outer key of NMAC-MD5 with *KR1*. We describe here how the inner key recovery attack of [6] can be extended to a full key recovery attack. The same characteristic [7] 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 $25 \times 2^{42} \approx 2^{47}$ queries. For each query the first word needs to be controlled, hence requires $25 \times 2^{42+32} \approx 2^{79}$ computations. Using this information, the full key can be guessed with $2^{128-25} = 2^{103}$ trials. 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^{42} queries per bit without additional overhead.

4.3 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, where a speed up factor 2^{30} compared to *KR1* is achieved.

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, wether 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.

A detailed description of *KR2* as well as the non-random properties needed for the compression function to make it more efficient than *KR1* are given in Appendix A.

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 characteristic with probability 2^{-148} from [10] can be used. Using *KR2* as proposed in this paper, key recovery is faster than brute force trials. Using 2^{153} queries, we recover 4 bits of key information. Hence the overall cost when using *KR2* for key recovery is 2^{156} which is more than 2^{30} times faster in this setting than *KR1* and hence slightly faster than brute force search.

5 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.

5.1 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 4 and Table 5. Table entries are either compared to previous results, or are new results for variants with more steps, while success rates of attacks are equalized.

On truncation. We also tackle the issue of truncation, which is (based on [16]) 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 of [16] that truncating helps against certain attacks but is a specific property of the newly devised attacks. In fact, if the output is further truncated than noted, the attack is stopped.

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

Table 4. 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	[9]
HMAC-SHA-1	34 (0-33)	forgery	2^{34}	64	[6]
HMAC-SHA-1	34 (0-33)	forgery	2^{32}	64	this paper
HMAC-SHA-1	37 (20-56)	forgery	2^{65}	96	this paper
	steps	**distinguisher**	data	truncation	source
HMAC-SHA-1	43 $(00-42)$	rectangle d.	$2^{154.9}$	160	[9]
HMAC-SHA-1	50 $(00-49)$	rectangle d.	$2^{153.5}$	160	this paper
HMAC-SHA-1	53 $(20-72)$	differential d.	$2^{97.5}$	128	this paper
HMAC-SHA-1	61 $(19-79)$	related-key differential d.	2^{100}	128	this paper

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).

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. However, if the goal is to distinguish NMAC/HMAC instantiated with a PRF from NMAC/HMAC instantiated with an actual hash function, the birthday bound does not apply [9]. Hence, as listed in Table 4, the new distinguishing attacks can cover more steps than the new forgery attacks.

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 4 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.

Key recovery. In Table 5 we summarize the new attacks that recover the full key of NMAC when observing a number of text/MAC pairs. Both attacks require related outer 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.

Table 5. Summary of key recovery attacks

	type	steps	data	offline	truncation	source
NMAC-MD5	inner rel. key	all (0-63)	2^{47}	2^{47}		[6]
HMAC-SHA-1	inner key	34 (0-33)	2^{34}	2^{34}		[6]
NMAC-MD5	full rel. key	all (0-63)	2^{47}	2^{103}	64	this paper
NMAC-SHA-1	full rel. key	34 (0-33)	2^{153}	2^{156}	160	this paper
HMAC-SHA-1	inner key	34 (0-33)	2^{32}	2^{32}	64	this paper
NMAC-SHA-1	inner rel. key	61 (19-79)	2^{100}	2^{100}	128	this paper
HMAC-SHA-1	inner key	53 (20-72)	$2^{99.5}$	$2^{99.5}$	128	this paper

5.2 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 [8]. 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 *e-SPR* resistance, is sufficient. Hence 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 B. 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 2.3 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 [8].

6 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 as *e. g.* proposed in [4] 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 stand-alone 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[1] and in part by the Austrian Science Fund (FWF), project P18138.

References

1. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In Neal Koblitz, 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.

[1] 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.

2. Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Ronald Cramer, 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.
3. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology, 4(1):3–72, 1991.
4. Christophe De Cannière and Christian Rechberger. Finding SHA-1 Characteristics: General Results and Applications. In Xuejia Lai and Kefei Chen, 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.
5. Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, 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.
6. Scott Contini and Yiqun Lisa Yin. Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In Xuejia Lai and Kefei Chen, 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.
7. Bert den Boer and Antoon Bosselaers. Collisions for the Compressin Function of MD5. In Tor Helleseth, 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.
8. Shai Halevi and Hugo Krawczyk. Strengthening Digital Signatures via Randomized Hashing. In Cynthia Dwork, 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.
9. Jongsung Kim, Alex Biryukov, Bart Preneel, and Seokhie Hong. On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1 (Extended Abstract). In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2006.
10. Jiqiang Lu, Jongsung Kim, Nathan Keller, and Orr Dunkelman. Differential and Rectangle Attacks on Reduced-Round SHACAL-1. In Rana Barua and Tanja Lange, 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.
11. Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. The Impact of Carries on the Complexity of Collision Attacks on SHA-1. In Matt Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Pre-Proceedings*, 2006.
12. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. Available online at `http://www.cacr.math.uwaterloo.ca/hac/`.
13. D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen. HOTP: An HMAC-based One Time Password Algorithm. Informational RFC 4226, December 2005.
14. 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`.
15. Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *LNCS*, pages 78–95. Springer, 2005.
16. Bart Preneel and Paul C. van Oorschot. MDx-MAC and Building Fast MACs from Hash Functions. In Don Coppersmith, 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.
17. Christian Rechberger and Vincent Rijmen. On Authentication with HMAC and Non-Random Properties. In Sven Dietrich, editor, *Proceedings of Financial Cryptography 2007, Trinidad and Tobago, February 12-15, 2007*, to appear in LNCS.

18. Xiaoyun Wang. What's the Potential Danger Behind the collisions of Hash Functions, 2005. ECRYPT Conference on Hash Functions, Krakow, available via `http://ecrypt.eu.org/stvl/hfw/`.
19. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, 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.
20. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, 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.
21. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, 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.
22. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Victor Shoup, 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.
23. Hongbo Yu, Gaoli Wang, Guoyan Zhang, and Xiaoyun Wang. The Second-Preimage Attack on MD4. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, 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, 1993.

A Details on the new key recovery method *KR2*

A.1 General description and difference to *KR1*

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

$$q = \Pr(h(k, m) + h(k, m + \alpha) = 0 \mid k \in \mathcal{K}, \mathtt{m} \in \mathcal{M}). \tag{2}$$

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 $\mathcal{K}$.

$$q' = \Pr(h(k, m) + h(k, m + \alpha) = 0 \mid k \notin \mathcal{K}, \mathtt{m} \in \mathcal{M}). \tag{3}$$

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. 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 [9, 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. Details can be found in Appendix A.2.
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

$$p_{\mathcal{K}} \cdot log_2(p_{\mathcal{K}}) + (1 - p_{\mathcal{K}}) \cdot log_2(1 - p_{\mathcal{K}}) \tag{4}$$

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.

A.2 ϵ_1

ϵ_1 can be derived as follows:

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

A.3 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 4.2 is of that type.

B Characteristics

For the characteristics, we adopt the notation introduced in [4]. 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 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 [4].

Table 6. Type 2 characteristic with probability 2^{-31} used for the 34-step (0-33) attack

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	----------		

Table 7. 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 8. 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 9. 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 10. 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 11. 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 12. 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