Note on Distinguishing, Forgery, and Second Preimage Attacks on HMAC-SHA-1 and a Method to Reduce the Key Entropy of NMAC

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. The first distinguishing, forgery and second preimage attacks on step-reduced HMAC-SHA-1 have recently been presented by Kim *et al.* In this note we report on ongoing work to improve their data complexity and present new attacks on HMAC-SHA-1 covering more steps. Additionally, we show how a collision-based technique can be used to reduce the key entropy of NMAC-SHA-1. Finally we comment on the applicability of the used techniques for analyzing a recent randomized hashing proposal.
We expect the improvements as well as the new key entropy reduction technique to be applicable to HMAC and NMAC instantiated with other hash functions of the MD4 family as well.

1 Introduction

HMAC [1] is a widely used message authentication code (MAC). In the pioneering work of Kim *et al.* [5] first distinguishing attacks, forgery and second preimage attacks on HMAC instantiated with full or step-reduced MD4, MD5, HAVAL, SHA-0 and SHA-1 are presented. Clearly, the case where HMAC is instantiated with SHA-1 is the most relevant one in practice since HMAC-SHA-1 is heavily used. Even more so, since NIST supports HMAC-SHA-1 also after 2010, whereas SHA-1 as a hash function will not be supported by NIST anymore [7].

We present improvements of this work by lowering the data complexity and extending the attack to more steps of HMAC-SHA-1. *This note is not meant to be self-contained.* For an introduction and description of the general attacks and differential attacks, we refer the reader to section 1-3 of [5]. Unless stated otherwise, we also adopt definitions and notation of [5].

In Section 2 of this note, we outline the idea behind the improvements. These improvements are subsequently applied to distinguishing attacks, forgery attacks, and second preimage attacks on HMAC-SHA-1 with a reduced number of steps in sections 3, 4, and 5, respectively. In Section 6 we briefly describe a new way to reduce the key entropy of NMAC using a chosen message attack, and apply it to step-reduced NMAC-SHA-1. We comment on the applicability of our results on a randomized hashing proposal in Section 7. Finally, we summarize and conclude in Section 8.

2 Improving the data complexity of existing attacks

Kim *et al.* present two novel types of distinguishers for HMAC. Differential distinguisher and rectangle distinguisher. As it will turn out in later sections that for HMAC-SHA-1, differential distinguishers are better suited than rectangle distinguishers, we will stick to that setting unless noted otherwise.

2.1 Add linear relations between message bits

The attacks by Kim *et al.* fix only the difference of the message pairs. Our approach is now to impose also conditions on the messages in order to obtain a higher probability for the differential. It was already observed in the early analysis of SHA [3] that some conditions for following the characteristic can be expressed as linear relations between the message bits. By choosing only messages that satisfy these relations, we reduce the degrees of freedom we have, but we improve the probability to find a collision, see also [6, Section 3].

Subsequently, we illustrate the improvement 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 distinguishers 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 useful relations 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.

As an example, we can lower the data complexity for distinguishing 34-step HMAC-SHA-1 using the same characteristic as presented in [5] from 2^{53} chosen messages to 2^{32} chosen messages. More details of this example are given in Section 4. For an overview of other results using this technique, refer to Table 3.

2.2 Add impact of less probable characteristics to improve the probability of the differential

In [5], it is noted that it is computationally infeasible to “sum the square of the probability of all differentials with message difference α”. Subsequently some experiments are mentioned to obtain a lower bound for this probability.

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
$0, 2, \ldots, 30$	f_{IF}	9	5
1	f_{IF}	7	5
31	f_{IF}	7	4
$0, 2, \ldots, 30$	f_{XOR}	6	4
1	f_{XOR}	3	2
31	f_{XOR}	4	3
$0, 2, \ldots, 30$	f_{MAJ}	9	4
1	f_{MAJ}	6	4
31	f_{MAJ}	7	4

To obtain better estimates for the probability of all differentials with message difference α, we derive better lower bounds for one differential. The techniques first presented in [6] prove to be useful. By considering less probable characteristics because of carries, an analytical method is derived to calculate a better lower bound for the probability of a differential in the SHA-1 compression function. As an example consider the forgery attack on 37-step HMAC-SHA-1 described in Section 4. By considering the better lower bounds using the methods described above, we expect the forgery attack to be successful already after 2^{66} instead of 2^{68} chosen messages.

3 Revisiting distinguishing attacks on HMAC-SHA-1

Kim *et al.* present two novel types of distinguishers for HMAC. Differential distinguisher and rectangle distinguisher. In our analysis it turns out that differential distinguishers are better suited than rectangle distinguishers for HMAC-SHA-1. Nevertheless, to facilitate comparison, we give our results for both types.

3.1 Differential distinguishers

We first consider differential distinguishers. In Table 7 we give a suitable 53-step characteristic with probability 2^{-98}. For a reasonable success rate of the distinguisher, we need $2^2 \cdot q^{-1}$ chosen messages [5], where q is the probability of the used differential. An estimate for the data complexity of a distinguishing attack on HMAC-SHA-1 reduced to 53 steps would hence be 2^{100} chosen messages. By allowing also less probably characteristics for the same differential (see Section 2.2), the total probability is improved by a factor of 3. Thus the final estimate for the required chosen messages is about $2^{98.5}$.

3.2 Rectangle distinguishers

In Table 6 we give a 50-step characteristic with probability 2^{-72} suitable for a rectangle distinguisher. If we consider only one characteristic, the feed-forward operation amounts for another factor of 2^{-6}. A very conservative estimate for the data complexity of a distinguishing attack on HMAC-SHA-1 reduced to 50 steps would hence be $2^{159.5}$ chosen messages.

Note that by removing 7 steps at the end of the characteristic, we can compare this distinguisher with the 43-step distinguisher presented in Kim *et al.* Instead of $2^{154.9}$ chosen messages, we need only $2^{132.5}$ which is an improvement by a factor of $2^{22.4}$. Allowing more characteristics and more differentials will give better lower bounds for their probability. However, we do not expect better results than using the differential distinguisher presented in Section 3.1.

4 Revisiting forgery attacks on HMAC-SHA-1

The characteristic used for the forgery attack on 34-step HMAC-SHA-1 is given in the first 34 steps of Table 6. The characteristic has a probability of 2^{-31}. Since the characteristic is optimized for 50 and not for 34 steps, we can still improve the probability. By rotating the characteristic by 2 bit positions to the right, we can improve the probability to 2^{-30}. Rotating is possible because of two reasons. Firstly, the linear code describing the SHA-1 message expansion is invariant with respect to word rotation. Secondly, the characteristic requires a linear behavior of the modular addition with respect to differences. Hence the probability is always > 0, but the position within the word might change the probability. As a result, 2^{32} chosen messages are enough for a forgery attack with reasonable success probability.

As with the distinguishing attacks, the attack works only for 50% of the key space, since one condition in the chaining variable input to h_2 needs to be fulfilled. Note that the same analysis also applies to the 34-step attack given by Kim *et al.*

Due to the improvements in data complexity, it is possible to describe a forgery attack for a (reduced) HMAC-SHA-1 with more steps. In Table 5, we give a suitable 37-step characteristic with probability 2^{-66} which results in a forgery attack using 2^{68} chosen messages. By allowing also less probably characteristics for the same differential (see Section 2.2), the total probability is improved by a factor of 4. Thus the final estimate for the required chosen messages is about 2^{66}.

5 Revisiting second preimage attacks on HMAC-SHA-1

For a second preimage attack on 53-step HMAC-SHA-1, we can reuse the characteristic presented in Table 7. However, since we do not have control over the signs of the differences, the probability is lower bounded by 2^{-153}. Again we gain a factor of 3 by allowing less probably characteristics for the same differential as well. Hence the probability to generate a second preimage when given a first preimage by this attack can be estimated to be $q = 2^{-151.5}$.

Note that the success probability of the second preimage attacks given above assume a single block (before padding). The longer the message, the higher the success probability of this second preimage attack. If k is the number of message blocks of the first preimage, the actual probability to find a second preimage is $q \cdot (k-1)$.

6 On key recovery attacks on NMAC

In this section we describe a new method to use chosen messages to reduce the key entropy of NMAC when instantiated with a specific cryptographic hash function. NMAC can be described as follows:

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

The proposed key recovery attack on NMAC consists of two phases:

1. Online phase: in a chosen-message scenario, the attacker asks for b (m,NMAC(m)) pairs under the same unknown key of length $2l$. Analyzing the results, c linear relations between bits of k_1 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 $b + 2^{l-c}$ is smaller than 2^l. Subsequently the online phase is described in more detail. Before that, some definitions are needed.

Let Γ be a set of linear relations between bits in k_1, and let p_Γ be the probability that a k_1 picked from a uniform distribution satisfies these linear relations. Likewise, let Δ 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_1 satisfies Γ and m satisfies Δ. We write

$$q = \Pr(h(k_1, m) + h(k_1, m + \alpha) = 0 \mid k_1 \in \Gamma, m \in \Delta). \tag{2}$$

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_1 does not satisfy the relations given by Γ.

$$q' = \Pr(h(k_1, m) + h(k_1, m + \alpha) = 0 \mid k_1 \notin \Gamma, m \in \Delta). \tag{3}$$

1. Collect b MAC pairs under the unknown key with chosen messages m and $m + \alpha$ where $m \in \Delta$.
2. If we observe at least one colliding MAC pair, then $k_1 \in \Gamma$ with probability $1-\epsilon_1$. If we do not observe a colliding MAC pair, then $k_1 \notin \Gamma$ 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 [5, 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 $k_1 \in \Gamma$, the possible key space is reduced by a factor p_Γ^{-1}. If $k_1 \notin \Gamma$, the possible key space is reduced by a factor $(1 - p_\Gamma)^{-1}$. For a given p_Γ the reduction of key entropy is hence

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

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 $(\alpha_i, \Gamma_i, \Delta_i)$. 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_1 (*i. e.* Γ) to be linearly independent.

It remains to be described how to find triples $(\alpha_i, \Gamma_i, \Delta_i)$ for specific cryptographic hash functions. Again we choose SHA-1. In the following, we show how characteristics used for other purposes in this paper can be used as a building block for key-recovery attacks.

6.1 Chosen Message Key Recovery Attack on 34-step NMAC-SHA-1

As an example of the approach described above, we use the 34-step characteristic given in Table 6 to describe an attack to recover the 320-bit key with a computational complexity of 2^{272}. Message difference α and Δ are already given by the XOR difference in the message and the linear relations needed to fix all the signs of the message differences in the expanded message. For Γ, we exploit the fact that the relations $A_{0,4} = 0$ and $A_{1,4} = 1$ need to hold. Hence, $p_\Gamma = 2^{-2}$. According to (4) we expect to reduce the key entropy by about 0.811 bits.

By slightly changing Δ we can have the same probability but relations $A_{0,4} = 1$ and $A_{1,4} = 0$ need to hold instead, whereas $A_{x,y}$ refers to bit y in state variable A in step x and $x < 1$ refers to the chaining input of the compression function of SHA-1. Both triples can be used to reduce the key entropy by 1.5 bits in total.

To generate more triples $(\alpha_i, \Gamma_i, \Delta_i)$, we simply rotate the characteristic which is for the same reason possible as explained in Section 4. By rotating the 34-step characteristic given in Table 6 we arrive at the probabilities shown in Table 2. We note that key recovery attacks against variants with more steps are possible.

By using these 32 different characteristics (having 64 triples in total), we estimate the data complexity to reduce the key entropy by 48 bits to be about $2^{43.7}$ chosen messages. The remaining work factor to recover the full 320-bit key is hence $2^{320} \cdot 2^{-48} = 2^{272}$ trials.

7 A Note on Randomized Hashing

We briefly discuss the applicability of our results to the randomized mode of operation of hash functions as described in [4] when used with SHA-1.

Table 2. Log_2 of probabilities of word-wise shifted variants of the 34 step characteristic

shift	probability q
0	-31
1	-37
$2, \ldots, 25$	-34
26	-36.42
$27, \ldots, 29$	-34
30	-30
31	-40

Table 3. Old and new results on HMAC-SHA-1. Table entries either compare to the recent results of Kim *et al.* with ours, or give new results for variants with more steps.

	steps	**distinguisher**	data
Kim *et al.*	43 (00 – 42)	rectangle d.	$2^{154.9}$
this paper	43 (00 – 42)	rectangle d.	$2^{132.5}$
this paper	50 (00 – 49)	rectangle d.	$2^{159.5}$
this paper	53 (20 – 72)	differential d.	$2^{98.5}$
	steps	**forgery**	data
Kim *et al.*	34 (0-33)	forgery	2^{53}
this paper	34 (0-33)	forgery	2^{32}
this paper	37 (20-56)	forgery	2^{66}
	steps	**2nd preimage**	probability
Kim *et al.*	34 (0-33)	differential d.	2^{-51}
this paper	34 (0-33)	differential d.	2^{-43}
this paper	53 (20-72)	differential d.	$2^{-151.5}$

Indeed, our results on the second preimage resistance of HMAC-SHA-1 as described in Section 5 are applicable here. This implies that SHA-1 reduced to 53 steps is not *e-SPR* resistant as required by the proof of security in [4].

8 Conclusions and Future Work

Kim *et al.* presented distinguishing and forgery attacks on HMAC instantiated with several hash functions, including reduced versions of SHA-1. In this ongoing work we improved upon their results by exploiting the fact that the characteristics impose simple, linear conditions on message and key. Secondly, by using additional characteristics, we can reduce the chosen plaintext requirements of the attacks. This leads to the results presented in Table 3 and Table 4. We conjecture that these improvements are applicable to HMAC when instantiated with older hash functions like SHA-0 or MD5 as well.

It turns out that for the purpose of forgery and second preimage attacks on HMAC-SHA-1, differential distinguishers are better suited than rectangle distinguishers. Note that all our results regarding distinguishing, forgery and second preimage attacks on HMAC-SHA-1 apply to NMAC-SHA-1 in exactly

Table 4. Summary of key recovery attack on NMAC-SHA-1

	steps	data	reduction
this paper	34 (0-33)	2^{43}	48 bits

the same way. For differential distinguisher and forgery attacks, a venue for future improvements could be the use of multi-block characteristics.

In Table 4 we summarize the results of the newly developed key recovery technique when applied to step-reduced HMAC-SHA-1. Note that stronger attack models might lead to additional improvements in all presented cases. Adaptive techniques for choosing messages as well as allowing related keys are possible.

Acknowledgements

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

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.
2. Christophe De Cannière and Christian Rechberger. Finding SHA-1 Characteristics. To appear at ASIACRYPT 2006, earlier version will be available via http://www.csrc.nist.gov/pki/HashWorkshop/program.htm.
3. 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, pages 56–71. Springer, 1998.
4. 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, pages 41–59. Springer, 2006.
5. 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. Cryptology ePrint Archive, Report 2006/187, 2006. `http://eprint.iacr.org/`.
6. 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.
7. NIST. NIST's Policy on Hash Functions, 2006. available online at `http://www.csrc.nist.gov/pki/HashWorkshop/NIST%20Statement/NIST_Policy_on_HashFunctions.htm`.

A Notation

For the characteristics, we use the notation introduced in [2]. In short, $'x'$ 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 und W_i the expanded message words. The values in the column $P_c(i)$ denote $-log_2(p_i)$, where p_i is the probability to be paid at each step.

B Characteristics

Table 5. Characteristic used for the 37-step 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 6. Characteristic used for the 50-step 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 7. Characteristic used for the 53-step 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	--------------------------------		