Breaking a New Hash Function Design Strategy Called SMASH*

Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Austria
{Norbert.Pramstaller, Christian.Rechberger,
Vincent.Rijmen}@iaik.tugraz.at

Abstract. We present a collision attack on SMASH. SMASH was proposed as a new hash function design strategy that does not rely on the structure of the MD4 family. The presented attack method allows us to produce almost any desired difference in the chaining variables of the iterated hash function. Due to the absence of a secret key, we are able to construct differences with probability 1. Furthermore, we get only few constraints on the colliding messages, which allows us to construct meaningful collisions. The presented collision attack uses negligible resources and we conjecture that it works for all hash functions built following the design strategy of SMASH.

Keywords: SMASH, hash functions, cryptanalysis, collision.

1 Introduction

A lot of progress has been made during the last 10 years in the cryptanalysis of dedicated hash functions such as MD4, MD5, SHA-0, SHA-1 [2, 4, 5, 10, 12]. In 2004 and 2005, Wang *et al.* announced that they have broken the hash functions MD4, MD5, RIPEMD, HAVAL-128, SHA-0, and SHA-1 [13, 14]. Due to these recent developments we will have to work on the design and analysis of new hash functions in the future.

A proposal for a new design strategy for dedicated hash functions, called SMASH, has been presented at FSE 2005 by Lars Knudsen [7]. SMASH is a hash function design-strategy that does not follow the structure of the MD4 family. As an example, two specific instances were presented: SMASH-256 and SMASH-512. SMASH-256 and SMASH-512 can be seen as alternatives to SHA-256 and SHA-512 proposed by NIST [9].

We present here a collision attack on SMASH that works independently of the choice that is made for the compression function in the hash function. The attack is based on an extension of the *forward prediction property* already observed in the design document. Furthermore, we exploit the absence of a secret key to construct differentials with probability 1. We are able to construct almost any

* The work in this paper has been supported by the Austrian Science Fund (FWF), project P18138.

B. Preneel and S. Tavares (Eds.): SAC 2005, LNCS 3897, pp. 233–244, 2006.
© Springer-Verlag Berlin Heidelberg 2006

difference in the chaining variables and we have only few constraints on the colliding messages. This fact allows us to produce meaningful collisions.

We present the collision attack on SMASH in three constructive steps with the goal to break the two specific instances SMASH-256 and SMASH-512. In order to explain the attack we also define two simple instances of SMASH, referred to as SMASH-ORD3 and SMASH-ORDy.

Firstly, we apply the attack on a simple instance of SMASH, referred to as SMASH-ORD3, by choosing the finite field element different than the one for SMASH-256 and SMASH-512. Secondly, we extend this attack to break all simple instances of SMASH, referred to as SMASH-ORDy, but show that it is not successful for SMASH-256 and SMASH-512 due to the maximum message length. Finally, a further extension of the collision attack leads to a collision for SMASH-256 and SMASH-512.

This article is structured as follows. We recall the most important aspects of the SMASH design method in Section 2. In Section 3, we introduce the underlying principles of the attack and apply it to the simple instances SMASH-ORD3 and SMASH-ORDy. We extend the attack to cover SMASH-256 and SMASH-512 in Section 4. We shortly present the most important aspects of SMASH-256 and give an example of a meaningful collision. Section 5, discusses some ideas about how to modify the design-strategy SMASH such that it is immune to the presented attack. We conclude in Section 6. In Appendix A we present the equations and results to produce a collision for SMASH-512.

2 The SMASH Design Method

We present here an overview of the hash function design strategy presented in [7]. Basically, we follow the notation of [7], except that we denote finite field addition by '+', and we stick to the convention of [3] to denote a difference by $h' = h + h^*$.

2.1 Definition of SMASH

Knudsen [7] proposes a new hash function model with a nonlinear compression function f based on a bijective n-bit mapping. Let $m = m_1, m_2, \ldots, m_t$ be the message input after MD strengthening [8], where each block m_i consists of n bits. The hash output h_{t+1} is computed as follows:

$$h_0 = f(\mathrm{iv}) + \mathrm{iv} \tag{1}$$

$$h_i = f(h_{i-1} + m_i) + h_{i-1} + \theta m_i \quad \text{for } i = 1, \ldots, t \tag{2}$$

$$h_{t+1} = f(h_t) + h_t \ . \tag{3}$$

Different to the design strategy of the MD4 family, SMASH applies the compression function f also to the initial value (1) and to the final hash computation (3). This is done in order to avoid pseudo-collision attacks, since the attacker does not have full control over the chaining variable h_0. Applying f also to the final hash computation should make it impossible to predict the final hash value.

The multiplication by θ in (2) is defined as an operation in the finite field $\mathrm{GF}(2^n)$. Note, that for this section and Section 3, θ is an arbitrary field element in $\mathrm{GF}(2^n)$ with the only restriction that $\theta \neq \{0, 1\}$ as mentioned in [7].

The structure of SMASH in (2) exhibits a *forward prediction* property as already described in [7]. Let h_{i-1}, h^*_{i-1} be two intermediate hash values with difference $h'_{i-1} = h_{i-1} + h^*_{i-1}$. Choose a value for m_i and compute $m^*_i = m_i + h'_{i-1}$. Then

$$h'_i = h_i + h^*_i = (1 + \theta)h'_{i-1} \ . \tag{4}$$

2.2 Comparing SMASH with a Block Cipher Based Hash Function

The SMASH design can be compared to a block cipher based hash function operating in the Matyas-Meyer-Oseas mode [11] as shown in Figure 1. In this mode the intermediate hash value is computed as follows:

$$h_i = E_{h_{i-1}}(m_i) + m_i \quad \text{for } i = 1, \ldots, t \ . \tag{5}$$

The underlaying block cipher $E_{h_{i-1}}(m_i)$ in (5) can be replaced by the term $f(h_{i-1}+m_i)+h_{i-1}$ defined in (2). This is a block cipher following the Even-Mansour construction [6]—more precisely, an Even-Mansour construction with key $K = K_1K_2 = h_{i-1}h_{i-1}$. The only difference between the SMASH design and a block cipher based hash function operating in the Matyas-Meyer-Oseas mode is that the message m_i is multiplied by θ prior to the addition to the chaining variable h_i.

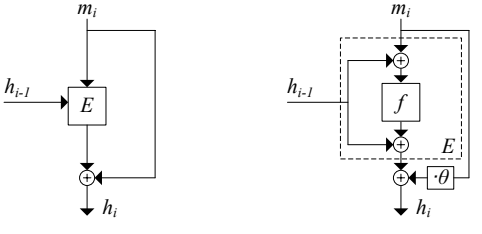

Fig. 1. The Matyas-Meyer-Oseas scheme (left) and the SMASH scheme (right)

3 Observation on the SMASH Design Method

We present here an observation on the design method explained in [7, Section 2]. The observation can be used to break the simple instances SMASH-ORD3 and SMASH-ORDy, but it does not break SMASH-256 nor SMASH-512. In Section 4, we explain how to extend this observation in order to break SMASH-256 and SMASH-512.

3.1 Target

In order to explain our attack, we first consider a simple instance of SMASH. The instance, referred to as SMASH-ORD3, differs from SMASH in the choice of the finite field element θ. We assume that we can choose a θ such that $(1 + \theta)$ has order 3, *i.e.* $(1 + \theta)^3 = 1$. Such a choice is not explicitly forbidden in [7].

3.2 Result

For describing the attack method in Section 3.3 we define the following variables (see also Figure 2):

$$\begin{aligned} x &\text{ an arbitrary 256-bit value} \\ f_1 &= f(m_1 + h_0) \\ f_2 &= f(m_2 + h_1) \\ f_3 &= f(m_3 + h_2) \\ a &= f_1 + f(m_1 + h_0 + x) + \theta x \ . \end{aligned}$$

The variable x defines an arbitrary 256-bit difference. $f_1 \dots f_3$ are the output values of f with message m_i and intermediate chaining variable h_{i-1} as input, *i.e.* without differences. a defines the difference in h_1. Based on these definitions, the following 4-block messages $m = m_1 m_2 m_3 m_4$ and $m^* = m_1^* m_2^* m_3^* m_4^*$ result in the same hash, for any value of z_1, z_2, z_3, and x. Note that a depends on both m_1 and x. In particular, if $x = 0$ then $a = 0$, *i.e.* $m = m^*$.

$$\begin{aligned} m_1 &= z_1 \\ m_2 &= z_2 \\ m_3 &= z_3 \\ m_4 &= z_1 + f_1 + f_2 + f_3 + \theta(m_1 + m_2 + m_3) \\ m_1^* &= z_1 + x \\ m_2^* &= z_2 + a \\ m_3^* &= z_3 + (1+\theta)a \\ m_4^* &= z_1 + f_1 + f_2 + f_3 + \theta(m_1 + m_2 + m_3) + (1+\theta)^2 a + x \end{aligned} \tag{6}$$

3.3 Description of the Attack Method

We describe here why we have a collision between the two 4-block messages m and m^* defined in (6). The attack is illustrated in Figure 2.

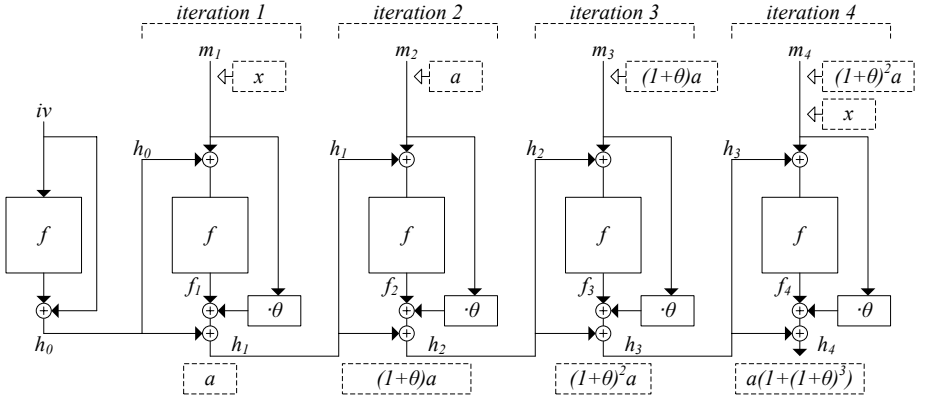

Fig. 2. The attack on SMASH-ORD3. The dashed rectangles denote differences.

The attack is an extension of the *forward prediction property* observed in [7]. It can be verified that the value a is the difference in h_1. We cannot predict the value of a, but we can of course easily compute it once we have chosen an arbitrary z_1 and x.

The basic idea of the attack is to control the propagation of the difference such that the input differences to the function f after the first iteration and prior to the last iteration (*iteration 2* and *iteration 3* in Figure 2) equal zero. In this case the nonlinearity of f does not have any impact on the difference propagation. For the last message block (*iteration 4*) we ensure that the difference $m'_4 = m_4 + m^*_4$ equals the difference $m'_1 = m_1 + m^*_1$ and that $f'_1 = f'_4$. This can be achieved as follows. By choosing the difference in the second message block equal to a, we make sure that the input difference to f equals zero (differences cancel out). Hence, we ensure that the difference in h_2 equals $(1+\theta)a$. Similarly, by choosing the difference in the third message block equal to $(1+\theta)a$, we ensure that the difference in h_3 equals $(1+\theta)^2 a$. This was already observed in [7].

Now we have to determine the last message block in each of the messages. We choose the last message block of the first message, m_4, in such a way that the input of f in the last iteration equals the input of f in the first iteration.

$$\begin{aligned} m_4 &= m_1 + h_0 + h_3 \\ &= m_1 + h_0 + f_3 + h_2 + \theta m_3 \\ &= m_1 + h_0 + f_3 + f_2 + f_1 + h_0 + \theta(m_3 + m_2 + m_1) \\ &= m_1 + f_3 + f_2 + f_1 + \theta(m_3 + m_2 + m_1) \end{aligned}$$

The last block of the second message, m^*_4, is selected in such a way that the difference in the last message block equals $(1+\theta)^2 a + x$. This choice ensures that the two inputs of f in the last iteration (*iteration 4*) equal the inputs in the first iteration (*iteration 1*). Consequently, the outputs of f will be the same as in the first iteration, hence they will have the same difference as in the first iteration: $a + \theta x$. Working out the equations, we see that the difference in h_4 becomes:

$$\begin{aligned} h'_4 = (f(m_1 + h_0) + h_3 + \theta m_4) + (f(m_1 + h_0 + x) + h_3 \\ + (1+\theta)^2 a + \theta(m_4 + (1+\theta)^2 a + x)) \\ = a + \theta x + (1+\theta)^3 a + \theta x \\ = a(1 + (1+\theta)^3)\ . \end{aligned} \tag{7}$$

Since we assumed a θ such that $(1+\theta)$ has order 3, the difference in (7), $h'_4 = a(1 + (1+\theta)^3)$, equals zero and we have produced a collision for our simple SMASH instance SMASH-ORD3. Due to the collision after the last iteration ($h'_4 = 0$), the final hash computation (3) has no impact on the result.

The attack on SMASH-ORD3 can be generalized to break the simple SMASH instances, referred to as SMASH-ORDy. The instances SMASH-ORDy are defined by choosing a θ such that $ord(1+\theta) = y$, where y is an arbitrary value. For a successful attack we then need at least $y+1$ message blocks without counting in the last message block that results from the MD strengthening. For instance,

for SMASH-ORD3 we have $y = 3$ and we have a collision in iteration $y + 1 = 4$. We will see in the next section that for the specific instances SMASH-256 and SMASH-512 this attack strategy does not work anymore. This is due to the number of maximum message blocks that can be hashed with SMASH-256 and SMASH-512.

4 Attacking SMASH-256 and SMASH-512

In this section we explain how the attack can be extended to break the proposed SMASH hash functions. After a description of the general attack strategy, we present some equations and solutions for the specific instance SMASH-256. Furthermore, we present an example for a meaningful collision. Equations and solutions for SMASH-512 are given in Appendix A.

4.1 SMASH-256

The hash function SMASH-256 is a specific instance of the design method SMASH. SMASH-256 is specified by setting $n = 256$, by defining the finite field $\mathrm{GF}(2^{256})$ via the irreducible polynomial $q(\theta)$,

$$q(\theta) = \theta^{256} + \theta^{16} + \theta^3 + \theta + 1, \tag{8}$$

and by defining the compression function f. Due to the chosen padding method, SMASH-256 can process messages with a bit length less than 2^{128}.

Even if the properties of f are not relevant for our attack, we shortly repeat them to give a basic understanding of the SMASH design strategy. The compression function f is composed of several rounds, called H-rounds and L-rounds:

$$f = H_1 \circ H_3 \circ H_2 \circ L \circ H_1 \circ H_2 \circ H_3 \circ L \circ H_2 \circ H_1 \circ H_3 \circ L \circ H_3 \circ H_2 \circ H_1 \ .$$

Both the H-rounds and the L-rounds take as input a 256-bit value and produce a 256-bit output. The underlaying operations are S-Boxes, some linear diffusion layers and variable rotations. The S-Boxes are based on the S-Boxes used for the block cipher Serpent [1]. An exact definition of the H-rounds and L-rounds can be found in [7].

4.2 Brief Description of the Attack

For the attacks on SMASH-ORD3 and SMASH-ORDy we assumed that we can choose a certain finite field element θ. This is not possible for the specific instances of SMASH. For SMASH-256 the finite field element θ is defined as a root of the irreducible polynomial $q(\theta) = \theta^{256} + \theta^{16} + \theta^3 + \theta + 1$, *i.e.* $q(\theta) = 0$. The irreducible polynomial for SMASH-512 is given in Appendix A. In order to show whether the previously described attacks on SMASH-ORD3 and SMASH-ORDy can be applied to SMASH-256 and SMASH-512, we have to compute the order of $(1 + \theta)$ for the specified θ. For SMASH-256, the order of $(1 + \theta)$ is $((2^{256} - 1)/5)$ and for SMASH-512 the order of $(1+\theta)$ is $(2^{512} - 1)$. Therefore, the

attack requires at least $(((2^{256}-1)/5)+1)$ message blocks for SMASH-256 and (2^{512}) message blocks for SMASH-512, respectively. As specified in [7], SMASH-256 can be used to hash messages of bit length less than 2^{128}. This corresponds to $(2^{120}-1)$ message blocks of 256 bits. SMASH-512 is specified for $(2^{247}-1)$ message blocks of 512 bits. Hence, the order of $(1+\theta)$ is for both hash functions larger than the maximum number of message blocks. This means, that we can still produce colliding messages but these messages are no longer valid inputs according to the SMASH-256 and SMASH-512 specification.

However, the attack technique can be generalized further. Previously, we extended the *forward prediction property* by considering message pairs that introduce a non-zero input difference x into f twice: once at the beginning and once at the end of the message. We can extend the property further by considering message pairs that introduce the difference x three or more times. Every time the input difference to f is non-zero, we make sure that the absolute values of the two message blocks equal the values in the first message blocks. Consequently, the output difference of f will be every time the same, namely $(a+\theta x)$. In this way, we can produce almost any desired difference in the chaining variable h_t. In order to find a collision, we want to construct a difference of the form $h'_t = a \cdot q(\theta) = a \cdot 0 = 0 \pmod{q(\theta)}$.

4.3 Equations

In this section, we introduce some notations and list the equations that need to be solved in order to construct pairs of messages that result in a specific difference in h_t. Without loss of generality, we will always work with messages that differ already in the first block, *i.e.* $m'_1 \neq 0$.

We define the following notation. Let d be a function defined for two input values only: $d(0) = 0$, and $d(x) = 1$. Let m'_i denote the difference in message block i. Let $\delta_1 = 1$ and let δ_i with $1 < i \leq t$, be defined as follows:

$$\delta_i = d(m'_i + \sum_{j=1}^{i-1} (1+\theta)^{i-j-1} a\delta_j) \ . \tag{9}$$

Then it is possible to construct two t-block messages with the differences defined by (9), such that the difference in h_t has the following value

$$h'_t = a \sum_{i=1}^{t} (1+\theta)^{t-i} \delta_i \ . \tag{10}$$

The absolute values m_i can be determined as follows. The first block, m_1, can always be selected arbitrarily. If $\delta_i = 0$, then m_i can be selected arbitrarily. If $\delta_i = 1$ and $i > 1$, then m_i has to be equal to $h_{i-1} + m_1 + h_0$.

4.4 Solutions for SMASH-256

The field polynomial $q(\theta)$ can be written as follows:

$$\theta^{256} + \theta^{16} + \theta^3 + \theta + 1 = 1 + (1+\theta)^2 + (1+\theta)^3 + (1+\theta)^{16} + (1+\theta)^{256} \ . \tag{11}$$

Hence, the solution of (10) is given by $\delta_i = 1$ for $i = 1, 241, 254, 255, 257$ and $\delta_i = 0$ for all other $i \leq t = 257$. Given the δ_i, (9) can be solved for the differences m'_i. This gives:

$$\begin{aligned}
m'_1 &= x \\
m'_i &= (1+\theta)^{i-2}a, && 1 < i \leq 240 \\
m'_{241} &= x + (1+\theta)^{239}a \\
m'_i &= (1+\theta)^{i-2}a + (1+\theta)^{i-242}a, && 241 < i < 254 \\
m'_{254} &= x + (1+\theta)^{252}a + (1+\theta)^{12}a \\
m'_{255} &= x + (1+\theta)^{253}a + (1+\theta)^{13}a + a \\
m'_{256} &= (1+\theta)^{254}a + (1+\theta)^{14}a + (1+\theta)a + a \\
m'_{257} &= x + (1+\theta)^{255}a + (1+\theta)^{15}a + (1+\theta)^2 a + (1+\theta)a \ .
\end{aligned}$$

Here, x is an arbitrary 256-bit difference. All other differences are defined by the attack method. As explained above, 253 of the message blocks m_i can be chosen arbitrarily, while the remaining 4 are determined by the attack.

The above-defined differences produce the following difference in the chaining variable h_{257}:

$$h'_{257} = (1+\theta)^{256}a + (1+\theta)^{16}a + (1+\theta)^3 a + (1+\theta)^2 a + a \ .$$

It is clear that $h'_{257} = a \cdot q(\theta) = a \cdot 0 = 0$, and hence we have a collision after iteration 257.

4.5 SMASH-256: Example of Colliding Messages m and m^*

In this section we give an example of two messages[1], m and $m^* = m + m'$, that collide after 257 iterations.

Figure 3 shows the two colliding ASCII coded strings. Each message consists of 258 message blocks. Even if we have already a collision after iteration 257 (see also Table 1), we added an additional message block, $m_{258} = m^*_{258}$, containing the character '>' ($3e_{hex}$). We have chosen the two messages in this way, because the message blocks $m_2 \ldots m_{258}$ and $m^*_2 \ldots m^*_{258}$ are inside the end tag ($< / m_2 \ldots m_{258} >$) and hence are not displayed in a standard *HTML* viewer or web browser (*e.g.* Mozilla Firefox 1.0.3). Therefore, at first sight, only the two message blocks m_1 and m^*_1 are visible.

Using hex notation, Table 1 shows the input message blocks m_i and m^*_i for $i = 1, 241, 254, 255, 257, 258$, the initial chaining variables $h_0 = h^*_0 = f(\mathrm{iv}) + \mathrm{iv}$, the chaining variables h_{257}, h^*_{257}, h_{258}, and h^*_{258}, and the colliding outputs h_{259} and h^*_{259}.

As described in Section 4.4, the messages $m_2 \ldots m_{240}, m_{242} \ldots m_{253}$, and m_{256} can be chosen arbitrarily. In this simple example each of these message blocks contains only space characters (20_{hex}).

[1] For this simple example we omitted padding.

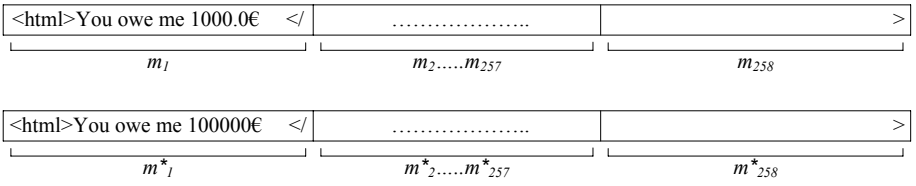

Fig. 3. ASCII coded strings m and m^*

Table 1. Colliding messages m and m^*

h_0=	55214e9e	237290c2	3ff782f7	c2073a8c	2105c5f1	6ccb0855	9c71b7c1	e7ecceac
h^*_0=	55214e9e	237290c2	3ff782f7	c2073a8c	2105c5f1	6ccb0855	9c71b7c1	e7ecceac
m_1=	3c68746d	6c3e596f	75206f77	65206d65	20313030	302e3030	80202020	20203c2f
m^*_1=	3c68746d	6c3e596f	75206f77	65206d65	20313030	30303030	80202020	20203c2f
				.....				
m_{241}=	346a6100	4e3cbc5b	f472d355	b41311b2	4b7df46d	e6b4028f	6aaf9c4d	97a6f169
m^*_{241}=	4afe2771	dd8507d9	a25082bc	dac25578	f34abb1c	5501e05e	d9874798	6aa679d3
				.....				
m_{254}=	60358467	cfde2276	534a4038	d3555d7e	576415d4	5c151dbb	7664ed09	f97bb393
m^*_{254}=	de2cdecd	6e323f7e	de8e653b	7d887168	f94cebb5	7370fc51	0e2e0226	8f1e25ba
m_{255}=	40088790	06da5567	eb2a1d6e	2869d96f	02fb791a	dc8799ca	0df2d9de	9dec9799
m^*_{255}=	f81b73fc	db0f8afe	28947e42	699822e0	6de2b3b5	0b5536fe	c3d1ebb0	7744d316
				.....				
m_{257}=	cc4a7c9c	9b4a99e1	8d275de9	3a44a2e7	4640484b	3cb2abb4	f1af679f	4e6e142f
m^*_{257}=	1a5d75eb	71ea319e	be76a60e	abc9278b	329ff04f	5e932f4d	cc04996a	9e6c4183
h_{257}=	ddc0b465	b42b5072	c34ad69d	b47c8e2c	30a36a7b	218a7bbe	99ffd185	831e8ddf
h^*_{257}=	ddc0b465	b42b5072	c34ad69d	b47c8e2c	30a36a7b	218a7bbe	99ffd185	831e8ddf
m_{258}=	20202020	20202020	20202020	20202020	20202020	20202020	20202020	2020203e
m^*_{258}=	20202020	20202020	20202020	20202020	20202020	20202020	20202020	2020203e
h_{258}=	da7c8fa1	4389e3c5	7299afdd	ad027de9	4c595315	c981c2f8	95390053	37c2fa00
h^*_{258}=	da7c8fa1	4389e3c5	7299afdd	ad027de9	4c595315	c981c2f8	95390053	37c2fa00
h_{259}=	2ffeac86	08bc1142	a3ddf493	6455bcd8	673dea34	c6365ec3	92b1bc79	15c1487e
h^*_{259}=	2ffeac86	08bc1142	a3ddf493	6455bcd8	673dea34	c6365ec3	92b1bc79	15c1487e

5 Discussion

In this section we discuss some observations on SMASH. Firstly, we propose a way how to change the SMASH design strategy such that it is not vulnerable to the presented attack. Secondly, we give some comments on block cipher based hash functions relaying on the SMASH design strategy.

5.1 Using Different Functions f

If a different f is used in each iteration, the attack described in this article seems not to work anymore. This is due to the fact that we expect the difference in the chaining variable h_i in iteration i, where $\delta_i = 1$, to be $(a + \theta x)$. If different functions are used for each iteration this cannot be ensured anymore and hence the presented attack is not successful. A simple method to modify SMASH could for instance be the addition of a counter value in each iteration. However, we did not further investigate these modifications and hence it should not be seen as a solution for this hash function design strategy. Another idea to modify SMASH such that it is immune against the presented attack is given in [7].

5.2 Block Cipher Based Hash Functions

In Section 2.2 we compared the SMASH design strategy with a block cipher based hash function. We have shown that the Matyas-Meyer-Oseas operation mode with a block cipher following the Even-Mansour construction is not secure.

6 Conclusion

We described a collision attack on SMASH-256 and SMASH-512. The attack works independently of the choice of the nonlinear compression function f and requires negligible computation power. We are able to construct meaningful collisions due to the fact that we have only few restrictions on the colliding messages. The attack is based on two observations. Firstly, the property of *forward prediction* which was described in [7]. Secondly, a differential attack on a hash function is easier than on a block cipher, because the attacker has control over the input values. If the attacker ensures that the two inputs to two different instantiations of the compression function are equal, then the two outputs (and hence the output difference) will also be equal in both instantiations.

If the compression function f would be different in every iteration, then it would not be possible to produce the same output difference twice.

Acknowledgements

We would like to thank Lars Knudsen for valuable conversations and for providing a reference implementation of SMASH-256.

References

1. Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A New Block Cipher Proposal. In Serge Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE 1998, Paris, France, March 23-25, 1998, Proceedings*, volume 1372 of *LNCS*, pages 222–238. Springer, 1998.
2. Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *LNCS*, pages 290–305. Springer, 2004.
3. Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology, 4(1):3–72, 1991.
4. 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.
5. Hans Dobbertin. Cryptanalysis of MD4. In Bart Preneel, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *LNCS*, pages 53–69. Springer, 1996.

6. Shimon Even and Yishay Mansour. A Construction of a Cipher From a Single Pseudorandom Permutation. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739 of *LNCS*, pages 210–224. Springer, 1991.
7. Lars R. Knudsen. SMASH - A Cryptographic Hash Function. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Proceedings*, volume 3557 of *LNCS*, pages 228–242. Springer, 2005.
8. 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/`.
9. National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002. Available online at `http://www.itl.nist.gov/fipspubs/`.
10. Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings to appear*, LNCS. Springer, 2005.
11. Bart Preneel. *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.
12. Vincent Rijmen and Elisabeth Oswald. Update on SHA-1. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *LNCS*, pages 58–71. Springer, 2005.
13. Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Xiuyuan Yu. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, August 2004. Preprint, available at `http://eprint.iacr.org/2004/199`.
14. 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.

A SMASH-512: Equations and Solutions

To determine the differences m'_i that produce a collision for SMASH-512 we use the same definitions and equations as presented in Section 4.3.

SMASH-512 is specified by setting $n = 512$ and by defining the finite field $\mathrm{GF}(2^{512})$ vie the irreducible polynomial $q(\theta)$

$$q(\theta) = \theta^{512} + \theta^8 + \theta^5 + \theta^2 + 1 \; . \tag{12}$$

The polynomial $q(\theta)$ can be written as follows:

$$q(\theta) = 1 + (1+\theta) + (1+\theta)^2 + (1+\theta)^4 + (1+\theta)^5 + (1+\theta)^8 + (1+\theta)^{512} \; . \tag{13}$$

The solution of (10) is given by $\delta_i = 1$ for $i = 1, 505, 508, 509, 511, 512, 513$ and $\delta_i = 0$ for all other $i \leq t = 513$. Given the δ_i, (9) can be solved for the differences m'_i. This gives:

$$
\begin{aligned}
m'_1 &= x \\
m'_i &= (1+\theta)^{i-2}a, && 1 < i \leq 504 \\
m'_{505} &= x + (1+\theta)^{503}a \\
m'_i &= (1+\theta)^{i-2}a + (1+\theta)^{i-506}a, && 505 < i < 508 \\
m'_{508} &= x + (1+\theta)^{506}a + (1+\theta)^2 a \\
m'_{509} &= x + (1+\theta)^{507}a + (1+\theta)^3 a + a \\
m'_{510} &= (1+\theta)^{508}a + (1+\theta)^4 a + (1+\theta)a + a \\
m'_{511} &= x + (1+\theta)^{509}a + (1+\theta)^5 a + (1+\theta)^2 a + (1+\theta)a \\
m'_{512} &= x + (1+\theta)^{510}a + (1+\theta)^6 a + (1+\theta)^3 a + (1+\theta)^2 a + a \\
m'_{513} &= x + (1+\theta)^{511}a + (1+\theta)^7 a + (1+\theta)^4 a + (1+\theta)^3 a + (1+\theta)a + a \ .
\end{aligned}
$$

Here x is an arbitrary 512-bit difference. All other differences are defined by the attack method. For SMASH-512, 507 of the message blocks m_i can be chosen arbitrarily, while the remaining 6 are determined by the attack.

The above-defined differences produce the following difference in the chaining variable h_{513}:

$$
\begin{aligned}
h'_{513} = {} & (1+\theta)^{512}a + (1+\theta)^8 a + (1+\theta)^5 a \\
& + (1+\theta)^4 a + (1+\theta)^2 a + (1+\theta)a + a \ .
\end{aligned}
$$

It is clear that $h'_{513} = a \cdot q(\theta) = a \cdot 0 = 0$, and hence we have a collision for SMASH-512 after iteration 513.