

Smashing SMASH

Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen

IAIK Krypto Group, Graz University of Technology, Austria
{norbert.pramstaller,christian.rechberger,vincent.rijmen}@iaik.tugraz.at

Abstract. We present a collision attack on SMASH. The attack uses negligible resources and we conjecture that it works for all hash functions built following the design method of SMASH.

Keywords: SMASH, hash functions, cryptanalysis, collision

1 Introduction

The hash function SMASH was presented as an example of a new design strategy for hash functions, different from the MD4-strategy [2]. We present here an attack that works independently of the choice that is made for the nonlinear bijective map 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 briefly recall the most important aspects of SMASH in Section 2. In Section 3, we introduce the underlying principles of the attack and apply it to a simplified variant of SMASH. In Section 4, we extend the attack to cover SMASH-256 and SMASH-512. We conclude in Section 5.

2 SMASH

We present here a quick overview of the hash function design strategy presented in [2], and the specific instance SMASH-256. We follow the notation of [2], except that we denote finite field addition by ‘+’, and we stick to the convention of [1] to denote a difference by $h' = h + h^*$.

Knudsen proposes a new hash function model based on a bijective n -bit mapping f . Let $m = m_1, m_2, \dots, m_t$ be the message input, where each block m_i is of n bits. The hash output h_{t+1} is computed as follows:

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

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

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

The properties of f are not relevant for our attack, hence we don't repeat them here. Multiplication by θ is defined as an operation in the finite field $\text{GF}(2^n)$.

This structure exhibits a ‘forward prediction’ property [2]. 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

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

The function SMASH-256 is a specific instance of this general model. It is specified by setting $n = 256$, by defining the finite field via the irreducible polynomial $q(\theta)$,

$$q(\theta) = \theta^{256} + \theta^{16} + \theta^3 + \theta + 1 , \tag{5}$$

and by defining the mapping f .

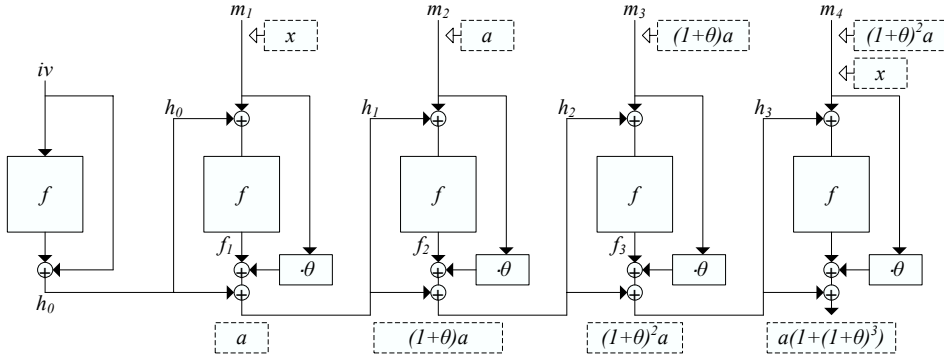


Fig. 1. The attack on a SMASH variant. The dashed rectangles denote differences.

3 Observation on the design method

We present here an observation on the design method explained in [2, Section 2]. The observation can be used to break SMASH-like hash functions, but it doesn't break SMASH-256 or 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 simplified variant of SMASH. The variant 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. Such a choice is not explicitly forbidden in [2].

3.2 Result

Define the following variables.

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

The following 4-block messages $m = m_1 m_2 m_3 m_4$ all 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 $a = 0$.

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

3.3 Description of the method

We describe here why we have a collision between two 4-block messages of the type (6): one message with $x = 0$ ('the first message') and one with $x \neq 0$ ('the second message'). The attack is illustrated in Figure 1.

The attack is an extension of the 'forward prediction' property observed in [2]. 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 fixed z_1 and x .

By choosing the difference in the second message block equal to a , 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 $(1 + \theta)^2a$. This was all already observed in [2].

Now we have to determine the last message block in each of the messages. We choose the last message block of the first message 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 is selected such that the difference in the last message block equals $(1 + \theta)^2a + x$. This choice ensures that the two inputs of f in the last iteration equal the inputs in the first iteration. 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:

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

Since we assumed that $(1 + \theta)$ has order 3, the difference (7) equals zero and we have produced a collision for our SMASH variant.

4 Attacking SMASH

In this section we explain how the attack can be extended to break the proposed SMASH functions. After a description of the general idea, we present some equations and solutions.

4.1 Brief description

The attack of the previous section can be extended to break all SMASH variants where the order of $(1 + \theta)$ is one below the maximum message length (in blocks). If the order is larger, we can still produce colliding messages, but these messages are no longer valid inputs according to the SMASH specifications. It turns out that both SMASH-256 and SMASH-512 are safe, because the order of $(1 + \theta)$ equals $(2^{256} - 1)/5$ for SMASH-256, and $2^{512} - 1$ for SMASH-512.

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 ($a + \theta x$). In this way, we can produce almost any desired difference in h_t . In order to find a collision, we want to cause a difference of the form $h'_t = a \cdot q(\theta) = a \cdot 0 = 0 \pmod{q(\theta)}$.

4.2 Equations

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

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:

$$\delta_i = d(m'_i + \sum_{j=1}^{i-1} (1 + \theta)^{i-j-1} a \delta_j) . \quad (8)$$

Then it is possible to construct two t -block messages with the differences defined by (8), such that the difference in h_t has the following value

$$a \sum_{i=1}^t (1 + \theta)^{t-i} \delta_i . \quad (9)$$

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 $h_{i-1} + m_1 + h_0$.

4.3 Solutions

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

$$\theta^{256} + \theta^{16} + \theta^3 + \theta + 1 = (1 + \theta) + (1 + \theta)^2 + (1 + \theta)^3 + (1 + \theta)^{16} + (1 + \theta)^{256} . \quad (10)$$

Hence, the solution of (9) 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 , (8) can be solved for the differences m'_i . This gives:

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

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

5 Conclusion

We described an attack on SMASH. The attack works independently of the choice of the nonlinear function f and requires negligible computation power. The attack is based on two observations. Firstly, the property of ‘forward prediction’ which was described in [2]. 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 nonlinear function are equal, then the two outputs (and hence the output difference) will also be equal in both instantiations.

If the f function would be different in every iteration, then it would not be so easy to produce the same output difference twice. A carefully designed variation of f over the iterations might make our attack more difficult. If the function would depend on a secret key, as in a block cipher, then producing the same output difference twice would be infeasible.

m_{230}^*	=	cc436f69	2aaafa0c	28881b02	d7e0e4b6	ad73913a	2f966773	a38b1f23	51d5505f
m_{231}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{231}^*	=	54c5b1bb	7fff0e15	79982d06	78212dda	f794b34f	70baa994	e49d2165	f27ff0e0
m_{232}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{232}^*	=	fd4fd2c6	8001123f	8aa8770a	8863766e	18bdd5d1	91cffabd	2da763af	16801121
m_{233}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{233}^*	=	07d0774a	80033640	9ff8991f	98a59ab3	29c67e72	b2500fc7	76e9a4f0	3b803363
m_{234}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{234}^*	=	087099de	80055ac0	a009ab20	a9eeafd4	7a4a8297	d6f01049	9b3aed11	4c8055a5
m_{235}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{235}^*	=	1891aa62	800fef40	e01afd61	fa33f07d	8edf87b8	7b1030db	ad4f3732	d580feee
m_{236}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{236}^*	=	29b3fead	801031c0	202f07a2	0e541086	936088c9	8d30516c	f7d15956	7e810333
m_{237}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{237}^*	=	7ad403f7	80305240	607108e7	12fc318a	b5a1995b	9750f3b5	1873ebfb	83830554
m_{238}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{238}^*	=	8f7d0412	8050f6c0	a0931928	3704529e	dee2abed	b9f114de	28943c0c	84850ffc
m_{239}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{239}^*	=	91860c3d	80f11b41	e1b52b79	590cf7a3	6327fc37	ca133d63	79bc4415	8d8f1004
m_{240}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{240}^*	=	b28b144c	81132dc2	22df7d8a	eb1518e4	a5680459	5e3547a5	8ac4cc3e	9691300c
m_{241}	=	bb9f291f	7296ad93	e9a62049	a3d2736d	d7efbf03	3d0b67f1	d44e71dc	f18e2f28
m_{241}^*	=	6c0315c0	f1a3dbd4	8ec7a6d6	9eed5a41	3857b3e9	df4aaf1f	4b03259e	4a3d7f3d
m_{242}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{242}^*	=	9d3ebf5a	b10b4f50	a08af2b1	356b3247	b1c83ac9	dec51780	ecc4536d	20107372
m_{243}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{243}^*	=	a743c1ee	d31dd1f1	e19f17d2	5fbd56c8	d2584f5b	634f3881	354cf5b6	60309597
m_{244}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{244}^*	=	e9c44232	75267212	22a13877	e0c7fb59	76e8d1ed	a5d14982	5fd51eda	a051beb9
m_{245}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{245}^*	=	3a4dc65d	9f6a9637	67e34899	21480deb	9b397236	ee73da86	e07f236f	e0f2c3cb
m_{246}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{246}^*	=	4ed74aec	a1bfba59	a825d9aa	63d8163d	ad4b965a	32946f8b	208165b0	2117445c
m_{247}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{247}^*	=	d379df34	e2c0ceeb	f86e6aff	a4683a46	f7dcbae	57bcb09c	6183aed0	6339cce4
m_{248}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{248}^*	=	758a615c	2741533c	08b2bf00	ecb84ecb	1865cf33	f8c5d1a5	a284f370	a54a552c
m_{249}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{249}^*	=	9e9fa3ef	69c3f544	19d7c100	35c8d35d	28ae5154	094e72ef	e78d1591	efdeff75
m_{250}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{250}^*	=	a3a1e43a	ba441fcd	2a784300	5e5975e7	79f2f3fc	1bd29731	28973eb3	3063019e
m_{251}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{251}^*	=	e4e22c4e	cecc2056	7e88c501	e2eb9e29	8a171404	2c77b953	79b943d5	50a502a2
m_{252}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{252}^*	=	2d2674d2	535460fb	83994f02	273ca27b	9e393c0d	7498cbf4	8acbc47f	f1ef07e6
m_{253}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{253}^*	=	776b9d7d	f5fca10d	84abd106	6945e68c	a24b4417	9da95c1d	9f5c4c81	1231082b
m_{254}	=	5cefbc2f	2cf4cbc9	b4b78b05	c4fbd3c0	1b50fb58	9a7063a1	3c0bb73f	4b003568
m_{254}^*	=	c5531ba8	32f128de	394bf80e	7f35f855	fd8d3761	3c958787	9def62bd	7d532d14
m_{255}	=	c17d180c	db400973	678c7e34	31bbc42c	16c770bb	d38de279	5ca12f5b	007beddf
m_{255}^*	=	8e230ab5	cd1af9d3	f8a09238	8fc3f1a1	bca10b06	c0ba80a1	f39ffe76	b64b4616
m_{256}	=	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
m_{256}^*	=	3478cdf0	0ebbc478	a85d4d05	b0a217a5	7faaa331	cd5ee9da	bc50dcdd	36947f16
m_{257}	=	1cdf6e29	58f91cfc	b14182e9	4675b348	7be44ef7	b247a7fa	9ee4abb3	0e03ad65
h_{257}	=	40563839	4b355074	49a655e6	97938ba6	fb1baba5	e5ba9d94	5a15ced5	55bf2c5e
h_{257}^*	=	686554f5	081ab981	23418980	110e325d	b62e5b8a	f73096f1	a7dd9266	5cfee75b
h_{257}^*	=	686554f5	081ab981	23418980	110e325d	b62e5b8a	f73096f1	a7dd9266	5cfee75b
SMASH	=	cc0b34fd	3821b07e	ebccec3	dc983778	cb0538f2	c5295b83	d0b0aa38	4325e171
SMASH*	=	cc0b34fd	3821b07e	ebccec3	dc983778	cb0538f2	c5295b83	d0b0aa38	4325e171