International Journal of Cryptology Research 1 (1): 103-114 (2009)

Crazy: A New 256-Bit Compression Function for Hash Algorithm

[1]Zhimin Li, [1]Bo Yang, [1]Lin Li, [1]Shihui Zheng, [1]Yixian Yang, [2]Zhihui Zhang

[1]Information Security Center State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications,
Beijing 100876, China
[2]Sony (China) Research Laboratory, Beijing 100080, China
Email: lizhmin@gmail.com

ABSTRACT

In this paper, we present a $\{0,1\}^{256} \times \{0,1\}^{512} \rightarrow \{0,1\}^{256}$ collision resistant compression function—Crazy, which can be used to design secure and efficient hash function. The inspiration of Crazy is from known attacks on current hash functions. Difference diffusion in the step function is so fast that the step function can be viewed as a random function after two steps. With the analysis we have done, we conjecture that it can resist all known cryptanalytic attacks applied to the compression functions. In addition, under the Merkle-Damgård iterative structure, the software performance of Crazy is 48% faster than that of SHA-256.

INTRODUCTION

A cryptographic hash function is usually composed by an iterative structure and a compression function, and a cryptographic iterative hash function $F(\cdot)$ with an n-bit output is assumed to have three security properties.

- Collision resistance: An attacker should not be able to find a pair of messages M and M' such that $F(M)=F(M')$ with less than about $2^{n/2}$ work.
- Preimage resistance: Given an output h in the range of the hash function, an attacker should not be able to find a message M such that $h=F(M)$ with less than about 2^n work.
- Second preimage resistance: Given one message M which is divided into about 2^k blocks for processing in F, an attacker should not be able to find a second message M' to satisfy $F(M)=F(M')$ with less than about 2^{n-k} work (This complexity was updated by NIST in their "SHA-3" competition call [21] and their special publication for "Recommendation for Using Approved Hash Algorithms" [20]).

The Merkle-Damgård iterative structure [3,18] is the most widely used for building a cryptographic hash function, as it gives a simple transformation that maintains the collision resistance property of the underlying compression function, such as MD5 [22] and SHA-1/256/512

[19] etc. Before 2004, there were few attacks on these known hash functions. However, in 2004 and 2005, a flood of cryptanalytic results [1, 2, 23-26] washed away most of the practical hash functions. Besides these cryptanalytic attacks applied to the compression function, there are also some generic attacks applied to the Merkle-Damgård iterative structure directly. These generic attacks aim to violate some properties other than collision resistance, and are attacks working on *n*-bit hash function with more than $2^{n/2}$ and much less than 2^n work. Examples of generic attacks are Joux multi-collisions [8], Kelsey and Schneier generic long-message 2^{nd} preimage attacks [9], Kelsey and Kohno herding attacks [10]. To resist these generic attacks, people should try to improve the Merkle-Damgård iterative structure or design some new structures. Considering the cryptanalytic attack applied to the compression function which can turn into an attack on the whole hash function, we'd better design a compression function with good diffusion effect. From the known cryptanalytic attacks applied to the compression functions, we found that all the attacked compression functions share the characteristic flaw of slow difference diffusion. Even for SHA-256, which is recommended for using by NIST, it is also much different from a random function within 7 steps (test result is listed in Figure 2. Then we want to know whether we could construct a compression function which diffusion effect surpasses these compression functions decisively. Designing a compression function used in hash function, we usually want to have a clear structure, and try to prove some mathematical theorems about their cryptographic properties. However, unfortunately, clean mathematical structure of such schemes can also help the attacker in his attempt to find an attack which is faster than exhaustive search. To resist attack, we have to use "crazy" combination of operations which belong to different domains and mix them such that difference in one bit would bring the change of many bits as soon as possible. Furthermore, we hope that these changes propagate so fast that the attacker can not find an effective attack. Thus, in this paper, we propose a new compression function—Crazy which has the following design goals:

- It should have a 256-bit output because the security is recommended as the computing power increases.
- Its structure should be resistant against known attacks [1, 2, 11, 14-16, 23-26].
- Under the Merkle-Damgård iterative structure, the performance should be better than that of SHA-256.

The rest of this paper is organized as follows. In Section 2, we describe the algorithm and give the design principles based on both security and efficiency. Security analysis is discussed in Section 3. Then, in Section 4, we compare the total number of operations and the software performance with

SHA-256. Summary remarks are given in Section 5. Finally, we give the process of Crazy in Appendix A.

DESCRIPTIONS AND DESIGN PRINCIPLES

Notations

There are some basic notations and functions used in Crazy, ⊞—addition mod 2^{32}, ⊕—XOR (exclusive or), ‖ —concatenation, $A^{>>>s}$—s-bit right rotation for a 32-bit word A.
Crazy uses four invertible logical functions, where each function operates on a 32-bit word which is represented as x. The result of each function is a new 32-bit word.

$$f_0(x) = x \oplus x^{>>>11} \oplus x^{>>>14}, f_1(x) = x^{>>>3} \oplus x^{>>>8} \oplus x^{>>>21}, \\ f_2(x) = x \oplus x^{>>>6} \oplus x^{>>>23}, f_3(x) = x^{>>>2} \oplus x^{>>>17} \oplus x^{>>>27}. \quad (1)$$

Before computation begins, the initial value $H^{(0)}$ must be set. Here we choose the first thirty-two bits of the fractional parts of the square roots of the first eight prime numbers. In addition, Crazy uses a sequence of twenty-four constant 32-bit words, K_0, K_1, … , K_{23}, in the message expansion and thirty-two constant 32-bit words, K_{24}, K_{25}, … ,K_{55}, in the step function. These words represent the first thirty-two bits of the fractional parts of the square roots of the following fifty-six prime numbers.

Structure of Step Function

Figure 1 depicts the step function of Crazy. In one step, two expansion words are used without intersection such that the difference introduced can not vanish in one step. In addition, we can found that Crazy has a different characteristic compared with other compression functions used in known hash functions. That is, the expansion word does not operate on any register directly. This feature will add the difficulty of difference control in the step function. For example, given a modular difference in the register a, if the attacker wants to eliminate the difference, he should modify W_{2j+1}, unless the register f has the same modular difference and the register h has the corresponding XOR difference, he will introduce new difference in the register f and h.

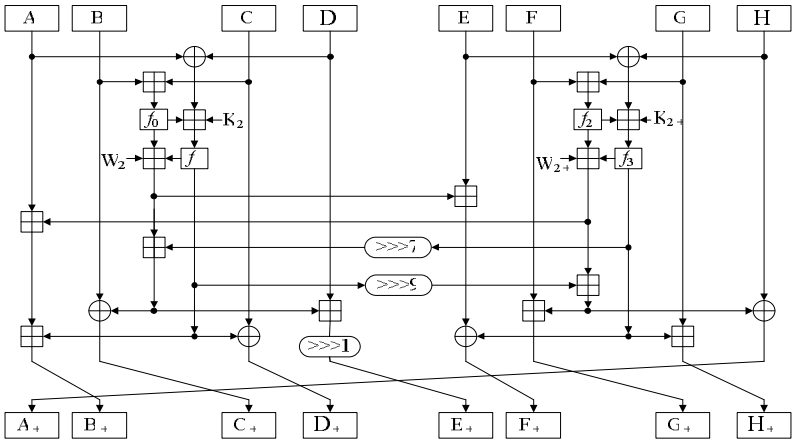

Figure 1: Computational graph of the step function

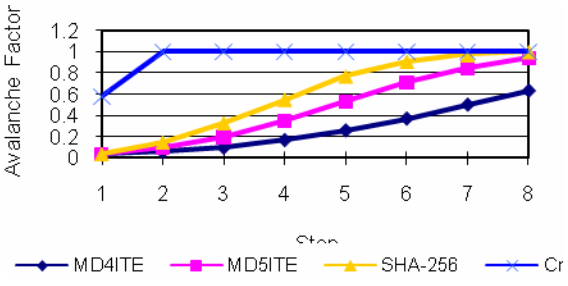

Figure 2: Step Dependence of the Avalanche Factor (When There is One Bit Input Difference)

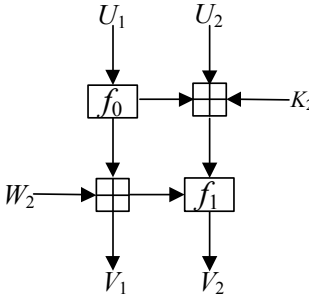

Figure 3: Computational graph of the FA structure

Through tests, we found that the step function of Crazy may be seen as a random function after two steps. We also compare the diffusion effect of Crazy with the compression functions of MD4, MD5 and SHA-256, and the test results are listed in Figure 2. (In [4], Daum used the avalanche factor ranged from 0 to 1, to compare the avalanche effect of step function of different hash functions. If the avalanche factor equals to 0/1, the function is seemed as a constant function/random function. In addition, "ITE" means the step function which Boolean function is $\mathrm{ITE}(x, y, z) = xy \oplus \bar{x}z$.) The reason of the fast diffusion of Crazy is the special structure used in the step function which is also the kernel of the step function; we call it Function-Addition (FA) structure, shown in Figure 3. Function f_0 and f_1 used in the FA structure are the functions depicted above and W_2 is an expansion word subblock, K_2 is a constant subblock. The main feature of the FA structure is

that, each output subblock depends on every input register subblock, but only the left output subblock depends on the message subblock, so difference in the register subblock can propagate successfully through the right line.

Message Schedule

Message schedule in the compression function of known hash functions usually have two kinds: one is message permutation, and the other is message expansion. To resist message modification technology, using message expansion seems to be better, as in a hash function with message expansion, the number of used times of each message are much more than that used in a hash function with message permutation. So we choose message expansion to do Crazy's message schedule.

Crazy has a "crazy" message expansion that Crazy uses step function to do message expansion such that each expansion word should be fully affected by every message. Then message modification will be very complex, even impossible. In the process of message expansion, we use initial value to help message expansion. The purpose is to strengthen the diffusion of the message and add the computation complexity of pseudo collision and multi-block collision. For the good diffusion effect of the step function, it is not necessary to have many expansion words like SHA family, and we think thirty-two expansion words are enough.

In the expansion course, message words are ordered following two permutations, $\pi_1 = 3_i + 1 (\mathrm{mod} 16), \pi_2 7_i (\mathrm{mod} 16), 0 \le i \le 15,$ and initial values are ordered following permutation $\rho = 5j \ (\mathrm{mod} 8), \ 0 \le j \le 7$. The permutations of message words and initial values are designed such that two message words/initial values that are used "closely" in the step function are far apart in the message expansion.

Rotation in Step Function

In Crazy, we have four invertible logical functions (1), and they have the same structure as $f_i = x^{>>>s_0^i} \oplus x^{>>>s_1^i} \oplus x^{>>>s_2^i}$, $i = 0,1,2,3.$ We consider all 4960 cases for s_0 , s_1 and s_2, and choose the one which can make the diffusion effect of each f and their combinations as well as possible. Besides these logical functions, there are three single right rotations (t_1=7, t_2=9, t_3=1) in the step function. The principles for choosing t_1 and t_2 are mainly from the points of difference diffusion. Given one bit difference of one registers, for each choice of t_1 (let t_2=0, t_3=0), we record the change of the output difference of the registers after proceeding two steps. At last, we choose t_1=7 and t_2=9 (similar tests like t_1) which are coprime with 32 and different from

s_j^i, s_2^i-s_1^i and s_1^i-s_0^i, i=0,1,2,3, j=0,1,2. The principles for choosing t_3 are from the points of cryptanalysis view. In the step function of Crazy, difference in the fore-step should be maintained to the next step, it is good from the points of diffusion view. However, it has a worst case that, if the corresponding registers which are combined before using in the FA structure have the same or negative differences, these differences might be neutralized before going into the FA structure. As a result, the output difference of this step is the same as that of the forward step. The simplest case is that, all the registers have one bit difference in their most significant bit. To solve this problem, we add a rotation on the computation line of the register *d*. Through test, we found that only one bit rotation is enough, so we choose t_3 =1.

SECURITY ANALYSIS OF CRAZY

As we propose a secure compression function, here we focus on the collision resistance of Crazy and consider all cryptanalytic attacks applied to the compression functions of known hash functions. Firstly, we will show the difference diffusion in Crazy.

Collision Analysis of Known Hash Functions

To derive a collision/near collision, different compression functions have different analysis methods. For MD4 Family (MD4, MD5, HAVAL, SHA-0/1 etc), the universal method is that, firstly, finding local collision without thinking about the details of the message schedule, in this process, all nonlinear components of the compression function are approximated by some suitable linear functions. Once a local collision is obtained, the attacker attempts to find a collision for the full hash function by taking into account the message schedule and the nonlinear behavior in the compression function. According to the properties of the step function, some techniques are used to improve the collision search methods, such as neutral bits used in the collision search for SHA-0/1 [1, 2], message modification techniques used for all these MD4 Family [23-26]. In the cryptanalysis of Reduced-Round Tiger [11, 15], there also exists local collision, but for its special message schedule, the main cryptanalysis tool is message modification technique. FORK-256 is a dedicated hash function that produces a 256-bit digest. The original version of FORK-256 was presented in the first NIST hash workshop and at FSE 2006 [6]. Some analyses have been done, and the most significant attack result is of Matusiewicz, Contini and Pieprzyk's [14]. They used the fact that function *f* and *g* used in the step function are not bijective and found collision of full FORK-256 with complexity of 2^{108}. In response to these attacks, the authors of FORK-256 proposed a new version of FORK-256 [7], which is supposedly resistant to these attacks. However,

Markku-Juhani O. Saarinen gave a novel and surprisingly simple analysis on the new FORK-256 [12]. He used the observations that each branch of the compression function uses each message word exactly once such that message words which are scheduled for the last steps do not affect all output words. And he gave a $2^{112.9}$ collision attack against the FORK-256 hash function both new and original versions.

These attacks above all have in common that the attacker has the ability to control difference diffusion in some special cases. The relatively slow diffusion of the step function gives the attacker chance to utilize the weak properties of the step function and the flaws in the message schedule fully. In the following, we will show the difference diffusion in Crazy. The good diffusion effect will make these attacks very difficult.

Difference Diffusion in Crazy

Without considering the message expansion, assuming that the attacker can modify message arbitrarily, we change addition into bitwise XOR operation in the step function of Crazy. When the attacker inserts one difference into the most significant bit of the first expansion word $W[0]$, Table 1 shows the change of the corresponding registers and the corresponding modification of the following expansion words. The numbers in the entries of the table denotes the bits in which the difference exists. Let $\Delta W[j]$ be the difference of $W[j]$, Δa^s (Δb^s, …, Δh^s) be the s^{th} step input difference of $a(b, \ldots, h)$.

TABLE 1: Differences Diffusion in Crazy

Step(s)	*0*	*1*	*2*	*1*	*2*
$\Delta W[2s]$	*31*			*1,7,9,10,12,14, 20,21,22,27,28, 29,30,31*	
$\Delta W[2s+1]$				*0,1,3,4,6,8,18, 19,22,25,28,29, 30,31*	
Δa^s			*0,1,3,4,6,8,18,19,22, 25,28,29,30,31*		
Δb^s			*4,5,6,8,9,10,12,17,23 ,27,29,30,31*		*0,4,5,8,10,12, 1318,23,26, 28*
Δc^s		*31*	*1,7,9,10,12,14,20,21, 22,27,28,29,30,31*	*31*	
Δd^s			*8,10,12,13,17,23,26, 28*		*8,10,12,13,17, 23, 26,28*
Δe^s		*30*	*0,6,8,9,11,13,19,20, 21,26,27,28,29,30*	*30*	

Δf^s		*31*	*4,5,6,8,9,10,12,20, 23,28,29*	*31*	*1,6,7,13,16,20, 21,22,23,29, 31*
Δg^s			*0,1,3,4,6,8,18,19,22, 25,28,29,30*		*31*
Δh^s			*4,5,6,8,14,28,29,30*		*4,5,6,8,14,28, 29,30*

In the forth column of Table 1, we use no message modification to control the differences diffusion, we can find that all the registers should be changed after two steps. Moreover, for the asymmetry of the combination of the registers, though the FA structure "XORing" the same bits into two output registers, XORing those registers in the next step ($a \oplus d$, $b \oplus c$, $e \oplus h$, $f \oplus g$) do not remove the non-linear contribution from the FA-structure or make differences disappear. And in the sixth column, we control the difference diffusion without thinking about the number of modified bits of the expansion words. Since Crazy has a special structure that no register is affected by message directly, five registers must be changed in the third step at least. Though people can control six register values (except for the register *d* and the register *h*) through modifying the corresponding expansion words, he should know that one word would affect three registers in one step simultaneously. Once he corrected the value of one register, other two registers must be changed.

In this linear variant of Crazy, we can find local collision using the properties of the logical function (using 0x*FFFFFFFF* as inserted difference for some messages). However, we do not know how to change it to a real local collision for Crazy. And we haven't found a local collision so far. Moreover, for Crazy's special expansion, people can not modify message so arbitrarily.

Invertibility of Crazy

From the analysis depicted above, we can found that Crazy has a well and "indestructible" diffusion effect. Unfortunately, mass and asymmetry combination of operations also make the step function of Crazy seem to be not invertible. Although we haven't found one step collision (for fixed message) so far, we cannot prove the invertibility of Crazy. However, this does not directly give rise to proper collision attacks. On one hand, through analysis, we found that the computational complexity of finding preimage of one step of Crazy is 2^{64}, which is too high to utilize. On the other hand, even if people can find one step collision for some register values and messages, using this one step collision in a real attack is still very hard. For a meet-in-the-middle attack, since the output values of the step function are all affected

by the input register values and two of them are only affected by the input register values in one step, the attacker cannot control the values of them arbitrarily through changing the values of the expansion words. For a pseudo collision attack, since the initial values are used to help message expansion, different initial values will result in different expansion words.

As the FA structure used in Crazy we spent more than two months to try all the analysis methods we could do, however we have little evolution. In addition, the fact that we use initial value to help message expansion can amplify the security of resisting multi-block collision. When the complexity of finding one block collision is too high, people consider two blocks and even more to reduce the complexity, and the complexity of a multi-block collision is not much larger than that of a near collision of one block. However, even if people can find near collision of one block in Crazy, the analysis on the second block should be more complex than the first block as the expansion words which used in the second block have already been changed. After a period of cryptanalysis on Crazy, we conjecture that Crazy can resistant against all known attacks up to now.

EFFICIENCY AND PERFORMANCE

In this section, we total the number of operations used in Crazy and compare it with the compression function of SHA-256. We also compare the software performance of Crazy under the Merkle-Damgård iterative structure with SHA-256.

Results are shown in Table 2.

TABLE 2: Compare Results of Number of Operations used and Software Realizationof Crazy and SHA-256

		Crazy		*SHA-256*	
Addition(⊞)		*512*		*600*	
Bitwise operation(⊕,∧,∨)		*392*		*1024*	
Shift(<<,>>)				*96*	
Rotation(<<<,>>>)		*364*		*576*	
		Mbps	*Cycle/Byte*	*Mbps*	*Cycle/Byte*
PM1.73G/504M	*VC++6.0*	*469.48*	*28.11*	*330.33*	*42.90*
	VS2005	*676.59*	*19.51*	*458.61*	*30.90*

SUMMARY

In this paper we have proposed a secure and software-efficient compression function. The main and novel features of Crazy are listed in the following:

- Using step function to do message expansion such that message modification is harder than before.
- To intensify the attack complexity, initial values are introduced to help the message expansion.
- Difference diffusion in the step function is so fast that message value can be completely diffused to all the registers after two steps.
- Expansion word does not operate on any register directly, so difference control is harder than before.

Our software performance results show that Crazy is 48% faster than that of SHA-256. So if people who want to have higher security, he can add appropriate expansion words. Crazy looks resistant against all the existing attacks based on the analysis we have done. Here we invite people to attack Crazy and will be thankful to receive the results (positive or negative) of any such attacks. Restricted to the length of paper, we do not give the source code of Crazy; people who have interesting may send us email.

REFERENCES

[1] Biham, E and R. Chen. 2004. Near Collisions of SHA-0. *Advances in Cryptology-CRYPTO 2004*, volume 3152 of LNCS, 290-305.

[2] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet and W. Jalby 2005. Collisions of SHA-0 and reduced SHA-1. *Advances in Cryptology-Eurocrypt 2005*, volume 3494 of LNCS, 36-57.

[3] I. Damgård. 1989. A Design Principle for Hash Functions. *Advances in Cryptology-CRYPTO 1989*, volume 435 of LNCS, 416-427.

[4] M. Daum. 2005. Cryptanalysis of Hash Functions of the MD4-Family. PHD thesis Bochum, Mai.

[5] H. Gilbert and H. Handschuh. 2003. Security analysis of SHA-256 and sisters. *Selected Areas in Cryptography 2003*, volume 3006 of LNCS, 175-193.

[6] D. Hong, D. Chang, J. Sung, S. Lee, S. Hong, J. Lee, D. Moon and S. Chee. 2006. A New Dedicated 256-Bit Hash Function: FORK-256. *Fast Software Encryption 2006*, volume 4047 of LNCS, 195-209.

[7] D. Hong, D. Chang, J. Sung, S. Lee, S. Hong, J. Lee, D. Moon and S. Chee. 2007. New FORK-256. *Cryptology ePrint Archive 2007/185*.

[8] A. Joux. 2004. Multicollisions in Iterated Hash Functions. *Advances in Cryptology-CRYPTO 2004*, volume 3152 of LNCS, 306-316.

[9] J. Kelsey and B. Schneier. 2005. Second Preimages on n-bit Hash Functions for Much Less than 2^n Work. *Advances in Cryptology-EUROCRYPT 2005*, volume 3494 of LNCS, 474-490.

[10] J. Kelsey and T. Kohno. 2006. Herding Hash Functions and the Nostradamus Attack. *Advances in Cryptology-EUROCRYPT 2006*, volume 4004 of LNCS, 183-200.

[11] J. Kelsey and S. Lucks. 2006. Collisions and Near-Collisions for Reduced-Round Tiger. *Fast Software Encryption 2006*, volume 4047 of LNCS, 111-125.

[12] M. O. Saarinen. 2007. A Meet-in-the-Middle Collision Attack Against the New FORK-256. *Indocrypt 2007*, volume 4859 of LNCS, 86-100.

[13] K. Matusiewicz, J. Pieprzyk, N. Pramstaller, C.Rechberger and V. Rijmen. 2005. Analysis of simplified variants of SHA-256. *WEWoRC*, volume 74 of LNI, pages 123-134.

[14] K. Matusiewicz, T. Peyrin, O. Billet, S. Contini and J. Pieprzyk. 2007. Cryptanalysis of FORK-256. *Fast Software Encryption 2007*, volume 4047 of LNCS, 19-38.

[15] F. Mendel, B. Preneel, V. Rijmen, H. Yoshida and D. Watanabe. 2006. Update on Tiger. *Indocrypt 2006*, volume 4329 of LNCS, 63-79.

[16] F. Mendel, N. Pramstaller, C. Rechberger and Vincent Rijmen. 2006. Analysis of step-reduced SHA-256. *Fast Software Encryption 2006*, volume 4047 of LNCS, 126-143.

[17] F. Mendel, J. Lano and B. Preneel. 2007. Cryptanalysis of Reduced Variants of the FORK-256 Hash Function. *CT-RSA 2007*, volume 4377 of LNCS, 85-100.

[18] R. Merkle. 1989. One way Hash Functions and DES. *Advances in Cryptology-CRYPTO 1989*, volume 435 of LNCS, 428-446.

[19] NIST. 2002. FIPS 180-2, Secure Hash Standard.

[20] NIST. 2007. SP 800-107, Recommendation for Using Approved Hash Algorithms (Draft).

[21] NIST. 2007. Announcing Request for Candidate Algorithm Nominations for a New Cryptograhic Hash Algorithm (SHA-3) Family.

[22] R. Rivest. 1992. The MD5 message-digest algorithm. Internet Request for Comment RFC 1321, Internet Engineering Task Force.

[23] X. Wang, Y. L. Yin, and H. Yu. 2005. Efficient collision search attacks on SHA-0. *Advances in Cryptology-CRYPTO 2005*, volume 3621 of LNCS, 1-16.

[24] X. Wang, Y. L. Yin and H. Yu. 2005. Finding collisions in the full SHA-1. *Advances in Cryptology-CRYPTO 2005*, volume 3621 of LNCS, 17-36.

[25] X. Wang, X. Lai, D. Feng, H. Chen and X. Yu. 2005. Cryptanalyisis of the hash functions MD4 and RIPEMD. *Advances in Cryptology-EUROCRYPT 2005*, volume 3494 of LNCS, 1-18.

[26] X. Wang and H. Yu. 2005. How to Break MD5 and Other Hash Functions. *Advances in Cryptology-EUROCRYPT 2005*, volume 3494 of LNCS, 19-35.

[27] H. Yoshida and A. Biryukov. 2005. Analysis of a SHA-256 variant. *Selected Areas in Cryptograph 2005*, volume 3897 of LNCS, 245-260.