Security Analysis of SHA-256 and Sisters[*]

Henri Gilbert[1] and Helena Handschuh[2]

[1] France Télécom R&D, FTRD/DTL/SSR
38-40 Rue du Général Leclerc, F-92131 Issy-Les Moulineaux
henri.gilbert@francetelecom.com
[2] GEMPLUS, Security Technologies Department
34 Rue Guynemer, F-92447 Issy-les-Moulineaux
helena.handschuh@gemplus.com

Abstract. This paper studies the security of SHA-256, SHA-384 and SHA-512 against collision attacks and provides some insight into the security properties of the basic building blocks of the structure. It is concluded that neither Chabaud and Joux's attack, nor Dobbertin-style attacks apply. Differential and linear attacks also don't apply on the underlying structure. However we show that slightly simplified versions of the hash functions are surprisingly weak : whenever symmetric constants and initialization values are used throughout the computations, and modular additions are replaced by exclusive or operations, symmetric messages hash to symmetric digests. Therefore the complexity of collision search on these modified hash functions potentially becomes as low as one wishes.

1 Introduction

A cryptographic hash function can be informally defined as an easy to compute but hard to invert function which maps a message of arbitrary length into a fixed length (m-bit) hash value, and satisfies the property that finding a collision, i.e. two messages with the same hash value, is computationally infeasible. Most collision resistant hash function candidates proposed so far are based upon the iterated use of a so-called compression function, which maps a fixed-length ($m+n$-bit) input value into a shorter fixed-length m-bit output value.

The most popular hash functions today are based on MD4 [20]. Following den Boer and Bosselaers [3], Vaudenay [23] and Dobbertin's work [7], it is no longer recommended to use MD4 for secure hashing, as collisions can now be computed in about 2^{20} compression function calls. In 1991, MD5 was introduced as a strengthened version of MD4. Although no collision has been found so far for MD5, pseudo-collisions were found on its compression function, so that MD5 is no longer considered as a very conservative hash function either [8, 9]. Other variants include RIPEMD, RIPEMD-128 and RIPEMD-160 [11, 19]. Attacks on

[*] This work is based on the result of an evaluation requested by the Japanese CRYPTREC project: http://www.ipa.go.jp/security/enc/CRYPTREC/index-e.html

M. Matsui and R. Zuccherato (Eds.): SAC 2003, LNCS 3006, pp. 175–193, 2004.

reduced versions of RIPEMD were published in [6, 10]. We call these functions the MD family of hash functions.

SHA, which also belongs to the MD family of hash functions, was introduced by the American National Institute for Standards and Technology and published as a FIPS standard in 1993. This early version is known as SHA-0. In 1994 a minor change to SHA-0 was made, and published as SHA-1 [14, 15]. The best attack known on SHA-0 is by Chabaud and Joux [4]. They show that in about 2^{61} evaluations of the compression function it is possible to find two messages hashing to the same value whereas a brute-force attack exploiting the birthday paradox would require about 2^{80} evaluations. The best known cryptanalysis reported on SHA-1 addresses the existence of slid pairs [22]. Finally a new generation of SHA functions with much larger message digest sizes, namely 256, 384 and 512 bits, called SHA-256, SHA-384 and SHA-512, was introduced in 2000 and adopted as a FIPS standard in 2002 [15]. As far as we know, the main motivation for introducing these new standard hash functions was to provide hash functions with security levels against collision search attacks consistent with the security levels expected from the three standard key sizes for the newly selected Advanced Encryption Standard (128, 192 and 256 bits) [16].

In this paper we study the security of these new functions against known attacks and report a very surprising property on a simplified version of these functions. For practical reasons, whenever our results apply to all three variants of SHA, we will denote these by SHA-2. For all other cases, the original name of the function will be used.

The rest of this paper is organised as follows. Section 2 briefly describes SHA-256, SHA-384, and SHA-512. Section 3 contains preliminary remarks on their main design features and a comparison with the corresponding features of SHA-1. Section 4 investigates the applicability of the currently known attacks of cryptographic hash functions to SHA-2. Section 5 shows that close variants of SHA-2 with modified constant values are not collision resistant, and sect. 6 concludes the paper.

2 Outline of SHA-256 and SHA-384/512

SHA-256, SHA-384, and SHA-512 belong to the MD family of hash functions. Since SHA-384 and SHA-512 are almost identical, we will describe both functions as a single algorithm SHA-384/512, and indicate their differences at the end of this Section. SHA-256 (resp. SHA-384/512) results from the iteration of a $256 + 512$-bit to 256-bit (resp. $512 + 1024$-bit to 512-bit) compression function. The hash computations are the following.

Padding: First the message is right-padded with a binary '1', followed by a sufficient number of zeros followed by a 64-bit suffix (resp. a 128-bit suffix) containing the binary length of the original message, so that the length of the resulting

```
$T_1 = h + \Sigma_1(e) + Ch(e, f, g) + K_t + W_t;$
$T_2 = \Sigma_0(a) + Maj(a, b, c);$
$h \;\; = g;$
$g \;\; = f;$
$f \;\; = e;$
$e \;\; = d + T_1;$
$d \;\; = c;$
$c \;\; = b;$
$b \;\; = a;$
$a \;\; = T_1 + T_2;$
```

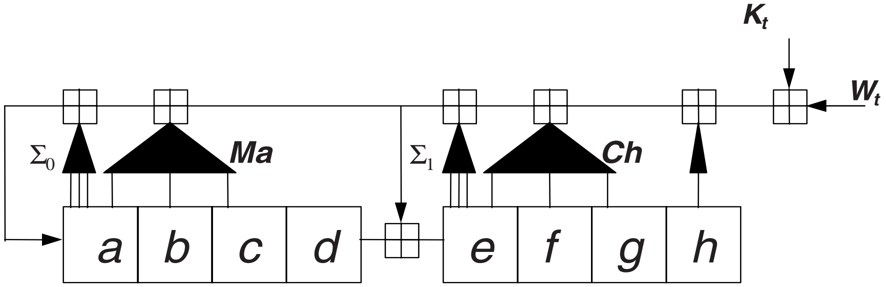

Fig. 1. State register update at each round of the compression function. The functions Ch, Maj, Σ_0 and Σ_1 do not depend on the round number, whereas K_t and W_t represent a constant and a message word the value of which depends on the round number t [15]

padded message becomes a multiple of 512 (resp. 1024) bits. The padded message is then cut into 512-bit blocks (resp. 1024-bit blocks). This form of padding is non-ambiguous and is an example of a valid padding for the Merkle-Damgård construction [5].

State Register Update: After the padding phase, the 8 state registers a, b, c, d, e, f, g, h are initialized to pre-determined 32-bit constants (resp. 64-bit constants) H_0 to H_7 (see [15] for a complete description) for the first message block, and to the current intermediate hash value for the following blocks. Next, 64 rounds (resp. 80 rounds) of the compression function are applied following the pseudo-code given below. Finally, the output value of the registers is added to the previous intermediate hash value according to the Davies-Meyer construction using addition (denoted '+') modulo 2^{32} (resp. 2^{64}) to give the new intermediate hash value.

In the case of SHA-256, the functions Ch, Maj, Σ_0 and Σ_1 operate on 32-bit input words, and produce the 32-bit words given by
$Ch(X, Y, Z) = (X \wedge Y) \oplus (\neg X \wedge Z);$
$Maj(X, Y, Z) = (X \wedge Y) \oplus (\mathrm{X} \wedge \mathrm{Z}) \oplus (\mathrm{Y} \wedge \mathrm{Z});$
$\Sigma_0(X) = ROTR^2(\mathrm{X}) \oplus ROTR^{13}(\mathrm{X}) \oplus ROTR^{22}(\mathrm{X});$
$\Sigma_1(X) = ROTR^6(\mathrm{X}) \oplus ROTR^{11}(\mathrm{X}) \oplus ROTR^{25}(\mathrm{X}).$

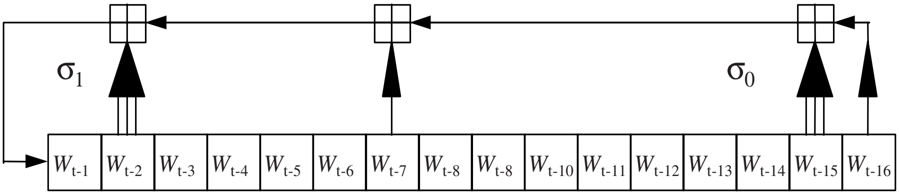

Fig. 2. Message schedule recurrence

In the case of SHA-384/512, the functions Ch, Maj, Σ_0 and Σ_1 operate on 64-bit input words, and produce the 64-bit words given by
$Ch(X, Y, Z) = (X \wedge Y) \oplus (\neg X \wedge Z)$;
$Maj(X, Y, Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$;
$\Sigma_0(X) = ROTR^{28}(\mathrm{X}) \oplus ROTR^{34}(\mathrm{X}) \oplus ROTR^{39}(\mathrm{X})$;
$\Sigma_1(X) = ROTR^{14}(\mathrm{X}) \oplus \mathrm{ROTR}^{18}(\mathrm{X}) \oplus ROTR^{41}(\mathrm{X})$.

Message Schedule: The 'message schedule' takes the original 512-bit message block (resp. the original 1024-bit message block) as input and expands these 16 32-bit words (resp. these 16 64-bit words) W_0 to W_{15} into 64 words W_0 to W_{63} (resp. into 80 words W_0 to W_{79}), one for every round of the compression function. This is done according to the following recurrence formula:

$$W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$$

where σ_0 and σ_1 represent linear functions (see also Fig. 2). In the case of SHA-256, the functions σ_0 and σ_1 operate on 32-bit input words, and produce the 32-bit words given by $\sigma_0(\mathrm{X})= ROTR^7(\mathrm{X}) \oplus ROTR^{18}(\mathrm{X}) \oplus SHR^3(\mathrm{X})$ and σ_1 $(\mathrm{X})= ROTR^{17}(\mathrm{X}) \oplus ROTR^{19}(\mathrm{X}) \oplus SHR^{10}(\mathrm{X})$. In the case of SHA-384/512, the functions σ_0 and σ_1 operate on 64-bit input words, and produce the 64-bit words given by $\sigma_0(\mathrm{X})= ROTR^1(\mathrm{X}) \oplus ROTR^8(\mathrm{X}) \oplus SHR^7(\mathrm{X})$ and $\sigma_1(\mathrm{X})= ROTR^{19}(\mathrm{X}) \oplus \mathrm{ROTR}^{61}(\mathrm{X}) \oplus SHR^6(\mathrm{X})$.

When all consecutive 512-bit message blocks (resp. all consecutive 1024-bit message blocks) have been hashed, the last intermediate hash value is the final overall hash value. The SHA-384 hash computations are exactly the same as those of SHA-512, up to the two following differences : the constants H_0 to H_7 used in SHA-384 are not the same as those used in SHA-512, and the SHA-384 output is obtained by truncating the final overall hash value to its 6 leftmost words.

3 Preliminary Remarks

3.1 Comparison with SHA-1

State Register Update Function: The overall structure of the 8-word state register (a, b, c, d, e, f, g, h) update performed at each round of the compression function of SHA-2 is close to the one of the 5-word state register (a, b, c, d, e) update performed at each round of SHA-1. However, one round of SHA-2 is more complex than one round of SHA-1: the Σ_0 and Σ_1 GF(2)-linear functions achieve a faster mixing than the circular rotations $ROTL^5$ and $ROTL^{30}$, the non-linear functions *Majority* and *Choice* are applied in each round whereas only one of the ternary functions *Choice*, *Majority* and *Xor* is applied in each SHA-1 round, and finally two register variables of SHA-2 are substantially modified at each round compared to only one for SHA-1. The SHA-2 round function is the same for all rounds except for the use of distinct constants K_t at each round, whereas SHA-1 involves four different types of round functions used in a subset of 20 consecutive rounds each. This lesser uniformity might represent a security advantage for SHA-1. But on the other hand equal constants K_t are used within each type of round function of SHA-1; unlike SHA-2, this makes SHA-1 vulnerable to Saarinen's sliding attack [22]. One can also notice that the number of rounds to state register length ratio, which represents the number of "full rotations" of the state register during each compression function computation, is much lower for SHA-2 than for SHA-1: its value is $64/8 = 8$ in the case of SHA-256 and $80/8 = 10$ in the case of SHA-384/512 instead of $80/5 = 16$ for SHA-1. The exact performance to security balance argumentation behind the substantial reduction of this ratio is unclear to us. This may look at first glance as a serious decrease of the security margin offered by SHA-2. On the other hand one can probably consider that this is at least partly compensated by the higher complexity of the round function (recall that two variables are updated in each round).

Message Schedule: Both the SHA-1 and the SHA-2 message schedules produce a recurring sequence of depth 16 initialized with the 16-word message block $M = M_0, M_1, \ldots, M_{15}$. However, unlike for SHA-1, the SHA-2 message schedule computations are not GF(2)-linear, because of the involvement of the '+' addition instead of '$\oplus$'. This makes the properties of the message schedule recurrence more difficult to analyze, because the set of possible difference patterns is no longer a linear code. The SHA-1 property following which (unlike in SHA-0) the recurrence relation mixes the various bit positions is strengthened thanks to the involvement of the bit rotations in σ_0 and σ_1 (which play a similar role as the $ROTL^1$ rotation of the SHA-1 recurrence) and also because of the diffusion effect introduced by the '+' addition.

3.2 Majority and Choice Functions

In this section we recall the elementary properties of the majority and choice functions as well as the modular addition operation [12]. Both the Choice and Majority functions operate on individual bits and are balanced on their respective input domains, as shown in their difference distribution table 1. The notation of table 1 is as follows: for every 3-bit input difference, a '0' denotes that the output difference is always zero, a '1' denotes that it is always one, and a '0/1' denotes that it is zero in half of the cases and one the rest of the time.

Table 1. Difference distribution table for a 3-bit input difference

X	Y	Z	$Choice$	$Majority$
0	0	0	0	0
0	0	1	0/1	0/1
0	1	0	0/1	0/1
0	1	1	1	0/1
1	0	0	0/1	0/1
1	0	1	0/1	0/1
1	1	0	0/1	0/1
1	1	1	0/1	1

For the subsequent sections, it is useful to note that both functions achieve a zero output difference (i.e. an internal collision) with average probability $\frac{1}{2}$ if exactly one of the three input differences is equal to 1.

Concerning the modular addition operation, one can easily see that if A and B differ in only the i-th bit, then with probability $\frac{1}{2}$ if a third word C is added to A and B, $(A + C)$ and $(B + C)$ also differ in only the i-th bit. The only special case here is when the difference is located in the most significant bit; in this case the carry bit does not propagate any difference, thus $(A + C)$ and $(B + C)$ differ also only in the most significant bit (with probability one) due to the modular reduction. This is already described in [12]. Thus on average, a one-bit difference before a modular addition does **not** propagate after the addition operation with probability $\frac{1}{2}$.

3.3 Sigma Functions

In this section we state some elementary properties of the functions Σ_0 and Σ_1 used in the state register update function and of the functions σ_0 and σ_1 used in the message schedule computations:

- **The GF(2)-linear mappings Σ_0 and Σ_1 are one to one**. In the case of SHA-256, this results from the fact that if 32-bit words are represented by polynomials over GF(2)$[X]/(X^{32}+1)$, then $\Sigma_0^{\{256\}}$ and $\Sigma_1^{\{256\}}$ are represented by multiplications by the polynomials $X^2+X^{13}+X^{22}$ and $X^6+X^{11}+X^{25}$, and these two polynomials are co-prime with $X^{32}+1=(X+1)^{32}$. Similarly, in the case of SHA-384/512, this results from the fact that the polynomials $X^{28}+X^{34}+X^{39}$ and $X^{14}+X^{18}+X^{41}$ are co-prime with the polynomial $X^{64}+1=(X+1)^{64}$.
- **The GF(2)-linear mappings σ_0 and σ_1 are one to one** (in order to check this property for SHA-256 and SHA-384/512, we computed the two 32×32 binary matrices (resp. the two 64×64 binary matrices) representing σ_0 and σ_1 and checked that the kernel of these matrices is restricted to the null vector.

These two observations tend to increase our confidence in the strength of SHA-2 versus SHA-1 as they provide for a much faster diffusion effect and, more importantly, no internal collisions can be achieved through either one of the Σ and σ functions.

4 Security of SHA-2 against Known Hash Function Attack Techniques

In this section we investigate the applicability of the currently known attacks of cryptographic hash functions to SHA-2.

4.1 Applicability of Chabaud and Joux's Attack Techniques

Chabaud and Joux's attack of SHA-0 is entirely differential in nature. It takes advantage of the absence of any mixing of the various bit positions in the SHA-0 message schedule expansion function and of its linearity to construct relatively low weight[1] differences on the W message schedule output which are likely to produce collisions during the SHA-0 compression.

This attack can be summarized as follows. First one considers the injection in one of the words W_t of a one-bit difference, and one identifies the corresponding corrective patterns, i.e. sets of differences in the subsequent words W_{t+i} that cancel with high probability the resulting differences in the state registers after a few rounds. Then we search for low weight sequences of 1-bit perturbative patterns satisfying the linear recurrence of the message schedule (and some extra technical conditions). Due to the structure of the SHA-0 message schedule, the difference pattern resulting from the superposition of these perturbative patterns and of the corresponding corrective pattern satisfies the linear recurrence. Therefore, numerous pairs of messages leading to one of these difference patterns are easy to construct and one of these pairs is likely to lead to a SHA-0 collision.

Following this attack, we investigate whether differential collisions may be obtained on SHA-2 faster than by exhaustive search. Using the differential properties of the non-linear functions shown in Sect. 3.2, we approximate each addition by an exclusive or operation with probability $\frac{1}{2}$, and the majority and choice functions by zero with probability $\frac{1}{2}$. We proceed in three steps:

- define a low weight perturbation and related corrective patterns;
- compute the probability that the corrective patterns produce a differential collision;
- produce heuristic evidence that these patterns may **not** be generated according to the SHA-2 message schedule.

Obviously, in order to obtain a minimum weight difference, the best strategy is to inject a one bit difference in a given message word W_i, and for each

[1] The difference weights encountered in this attack are higher by an order of magnitude than the extremely low difference weights encountered in Dobbertin's attacks.

Table 2. Hamming weight of a propagating difference for SHA-2

W	a	b	c	d	e	f	g	h
1	1	0	0	0	1	0	0	0
6	0	1	0	0	3	1	0	0
9	0	0	1	0	0	3	1	0
0	0	0	0	1	0	0	3	1
1	0	0	0	0	1	0	0	3
6	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0

consecutive round, to disable the propagation into the register A by appropriate corrective patterns of the next message words. Allowing for more than a single bit difference is not realistic as each Σ function automatically multiplies the Hamming weight of the difference by three in each step, and trying to match these bit locations using several initial difference bits implies a fatal decrease of the probability to obtain such differential collisions. Therefore we believe that no other strategy can provide a sufficiently low weight perturbation pattern, hence an acceptable overall collision probability. The pattern has been obtained in a straightforward manner by setting the following equalities: let W_i be the word containing the perturbative **one-bit difference**. Then we define the next eight word **differences** by: $W_{i+1} = \Sigma_1(W_i) \oplus \Sigma_0(W_i)$; $W_{i+2} = \Sigma_1(\Sigma_0(W_i))$; $W_{i+3} = 0$; $W_{i+4} = W_i$; $W_{i+5} = \Sigma_1(W_i) \oplus \Sigma_0(W_i)$; $W_{i+6} = 0$; $W_{i+7} = 0$; $W_{i+8} = W_i$.
This leads to the propagation of minimum weight differences in the 8 registers as shown in table 2. Explicit examples of such a difference propagation is given in Appendix A.

Fact 1. *Using corrective patterns with weight one, six and nine in the message words as shown in the W column of the above table gives rise to a differential collision pattern over 9 rounds.*

The next step is to evaluate the probability of the differential pattern. As we already mentioned, we approximate the addition operation by an exclusive or, and the non-linear functions by zero. Two relevant additions occur in every round per one bit difference in a message word. These are: the addition of T_1 and the register d to form the new register value e; the addition of T_1 and T_2 to form the new register value a. The probability of a carry bit appearing in one of these additions is bound by $\frac{1}{2}$ for each addition, thus we upper bound the overall probability for two additions per difference bit by $\frac{1}{4}$. There are a total of $1+6+9+1+6+1 = 24$ difference bits over the 9-round pattern. Hence the probability over all additions is upper bounded by $(2^{-24})^2 = 2^{-48}$.

As for the non-linear choice and majority functions, the average probability to obtain a zero difference is $\frac{1}{2}$ per difference bit. A total of 18 such difference bits occur in the non-linear functions over the 9 rounds; thus the overall associated probability is upper bounded by 2^{-18}.

Fact 2. *The overall probability associated to the 9-round differential collision pattern is upper bounded by* 2^{-66}*.*

In the third and last step, we provide evidence that these patterns may **not** be concatenated (as is the case for SHA-0) in order to form message words that follow the correct message schedule.

For SHA-256 suppose there is a block of 9 consecutive message words with differences defined as above (i.e. following the differential collision pattern). This block may not be followed or preceded by more than 15 zero difference message words. Clearly if 16 consecutive words in the message schedule are identical, then the whole message schedule is identical over the 64 words. If we apply the pattern twice, we can separate both patterns by a block of zero difference words of length at most 15. Therefore at most 15+9+15+9+15=63 difference message words may be defined. This shows that at least 3 different patterns must be combined to follow the correct message schedule.

If two of these patterns are applied, the probability to obtain a differential collision becomes lower than 2^{-132} whereas the complexity of a birthday attack on SHA-256 only represents 2^{128} computations on average.

Conclusion. *The original attack by Chabaud and Joux on SHA-0 does not extend to SHA-256.*

For SHA-384/512 suppose there is a block of 9 consecutive message words with differences defined as above (i.e. following the differential collision pattern). This block may not be followed by more than 7 pairs of identical message schedule output words. Let us show why.

Recall that the simplified message schedule (i.e. where every addition has been replaced by an exclusive or) is defined by:

$$W_t = \sigma_1(W_{t-2}) \oplus W_{t-7} \oplus \sigma_0(W_{t-15}) \oplus W_{t-16}$$

Then assuming that W_i to W_{i+8} represent a differential collision pattern and that W_{i+9} to W_{i+15} are equal to zero, the difference in the 16-th message word is defined by:

$$\begin{aligned} W_{i+16} &= \sigma_1(W_{i+14}) \oplus W_{i+9} \oplus \sigma_0(W_{i+1}) \oplus W_i \\ &= \sigma_0(W_{i+1}) \oplus W_i \\ &= \sigma_0(\Sigma_1(W_i) \oplus \Sigma_0(W_i)) \oplus W_i \\ &\neq 0 \text{ for a 1-bit difference in } W_i. \end{aligned}$$

Hence no more than 7 consecutive identical words may separate two consecutive differential collision patterns, with the original message block difference having at least one non-zero word.

Combining up to four different patterns, we can define at most 15 + 9 + 7 + 9 + 7 + 9 + 7 + 9 + 7 = 79 difference message words satisfying the message schedule recurrence. This shows that at least 5 different patterns have to be combined to follow the correct message schedule.

If four of these patterns are applied, the probability to obtain a differential collision becomes lower than 2^{-264} whereas the complexity of a birthday attack on SHA-512 only represents 2^{256} computations on average.

Conclusion. *The original attack by Chabaud and Joux on SHA-0 does not extend to SHA-384/512.*

4.2 Applicability of Dobbertin's Attack Techniques

The hash function attack techniques introduced by H. Dobbertin in [7, 8, 9, 10] exploit the extremely simple structure of the message schedule of hash functions such as MD4 and MD5. In these functions, the 16-words of the message block are just repeated in permuted order a small number r of times (namely 3 times for MD4 and 4 times for MD5). Dobbertin's attacks use pairs (M, M^*) of almost equal messages, up to a difference of Hamming weight 1 in only one of their 16 words. The attack strategy consists in controlling the diffusion of the resulting low weight (e.g. r-bit) message schedule output differences pattern through the hash function computations, in order for the resulting state registers differences after the $r-1$ first (16-step) rounds to be cancelled by the last message schedule difference encountered in the last (16-steps) round. Depending on the round considered, the control method consists either in differential techniques or in more sophisticated equation solving techniques.

Due to the more complex and conservative expansion method used in the message schedules of the SHA family of hash function, and in particular in the message schedule of SHA-2, Dobbertin's attacks do not seem applicable to these functions. More explicitly, the recurrence relation of the SHA-2 function (in particular the term $\sigma_0(W_{t-2})$ in this recurrence) ensures a fast and strong diffusion of any low weight difference in the message block M, and prevents any (M, M^*) pair of message blocks from resulting in a very low weight difference (e.g. 3, 4 or 5) at the message schedule expansion output.

4.3 Differential Attacks

Link between Differential Properties and Collision Resistance. In this section we investigate differential properties of the compression function. The idea behind this is that if it were possible to find any pseudo-collisions of the special form $compress(H, M) = compress(H', M)$ on the compression function of SHA-2, then the Merkle-Damgård security argument [5] could not be applied. In other words, the existence of such pseudo-collisions on the compression function of SHA-2 would represent an undesirable property.

In order to search for pseudo-collisions of this special form, it is convenient to view the compression functions of SHA-256 and SHA-384/512 as a block cipher[2] encrypting a 256-bit (or 512-bit) input $H = (a, b, c, d, e, f, g, h)$ under key $[W_0, \ldots, W_{63}]$ (or key $[W_0, \ldots, W_{79}]$) followed by the addition of the output with the input H according to the Davies-Meyer construction. If it were possible to predict the differential behavior of this block cipher, it might help find high probability differential characteristics such that the output difference compensates the input difference in the final Davies-Meyer addition, and thus to find pseudo-collisions of the form $compress(H, M) = compress(H', M)$. A similar approach has already been taken for DES-based hash functions in [18] and for SHA-1 in [12] in order to investigate the security of the underlying block cipher. No high probability differentials (and thus no partial collisions) could be found for SHA-1. However it should be mentioned that slid pairs (which result in related-key attacks for block ciphers) have since been discovered on MD-type hash functions used as block ciphers, including SHACAL-1 and MD5 [22].
Next we study the differential behavior of the block ciphers underlying SHA-2.

Search for Low Weight Differential Characteristics over a Few Rounds As in Section 3.1, we approximate each addition by an exclusive or operation with probability $\frac{1}{2}$, and the majority and choice functions by zero with probability $\frac{1}{2}$, using the differential properties of the non-linear functions shown in Sect. 3.2.

The most efficient differential characteristic over a few consecutive rounds we have identified relates to 4 rounds, and has a probability of 2^{-8}. See Appendix B for details.

While this characteristic does not concatenate over the whole 64 rounds, we can conclude that the best overall differential probability for SHA-256 appears to be lower than $2^{-8*16} = 2^{-128}$ which results in a work factor much higher than the complexity of a collision search for a 256-bit hash function. Thus a standard differential attack on the compression function is very unlikely to succeed.

While this characteristic does not concatenate over the whole 80 rounds, for SHA-512 we can conclude that the best overall differential probability appears to be lower than $2^{-8*20} = 2^{-160}$. As opposed to the case of SHA-256, this does not result in a work factor which is higher than the complexity of a collision search on a 512-bit or even a 384-bit hash function. However, it remains to be seen whether a global differential characteristic may be constructed from this property. Having in mind that we need 80-round characteristics with input and output differences that compensate each other in the final Davies-Meyer addition, there is no obvious way to extend this result to the hash function itself.

Search for Iterative Differential Characteristics We have also investigated iterative differential differential characteristics of non-negligible probability on

[2] In the case of SHA-256, this block cipher is the SHACAL-2 algorithm [13] recently selected by the European NESSIE project

a reduced number of rounds. For that purpose we have approximated the actual differential transitions associated with each round of the SHA-2 register update function by a linear function L of $\{0,1\}^{256}$ (resp. $\{0,1\}^{512}$). In order to identify candidate iterative differential characteristics over a restricted number r of rounds, we computed, for the 32 first values of r, a basis of the vector space of difference vectors $\delta = (\delta a,\ \delta b,\ \delta c,\ \delta d,\ \delta e,\ \delta f,\ \delta g,\ \delta h)$ which stay invariant under L^r. See Appendix B for details. It was observed that for the first values of r, all the 32-bit or 64-bit words of the vectors of the identified spaces of iterative differences are "periodic", of period 32, 16 or 8 (resp of period 64, 32 or 16) and thus the weight of all candidate iterative differential characteristics we identified using this method is a multiple of 8. Therefore, we believe that this approach does not provide high probability iterative differentials for SHA-2.

Consequently, we find the differential behavior of the compression functions of SHA-2 to be sound and believe it is highly unlikely that even pseudo-collisions will be found through this approach.

5 Weakness in SHA-2 Variants Using Symmetric Constants and Exor

In this Section we show that if some relatively slight variations are made in the SHA-2 specification, the resulting modified hash function is no longer collision-resistant. The considered variations consist in replacing all constants encountered in the algorithm by highly symmetric values and all additions modulo 2^n by the exclusive or operation. In order to simplify the discussion, we only describe the case of SHA-256, but the transposition to SHA-384/512 is straightforward.

Let us denote by Ω the set $\{0,1\}^{32}$, and by Ω' the set of all symmetric 32-bit words consisting of two equal 16-bit halves:

$$\Omega' = \{C \in \{0,1\}^{32} \mid \exists c \in \{0,1\}^{16}, C = c \| c\}.$$

Let us denote by SHA'-256 the SHA-256 variant obtained by replacing:

- the words $H_0^{(0)}$ to $H_7^{(0)}$ of the initial hash value $H^{(0)}$ by any 8 constant 32-bit words belonging to Ω';
- the constants K_0 to K_{63} involved in the hash computation by any 64 32-bit words belonging to Ω';
- the operation '+'(mod 2^{32} addition) in the hash computation by '$\oplus$';
- the shift operation $SHR^3(x)$ in the function $\sigma_0^{\{256\}}$ by the circular shift operation $ROTR^3(x)$ and the shift operation $SHR^{10}(x)$ of the function $\sigma_1^{\{256\}}$ by the circular shift operation $ROTR^{10}(x)$.

Now it is easy to see that if $x, y \in \Omega'$ then $x \oplus y \in \Omega'$ and that if $x, y, z \in \Omega'$, then $Ch(x,y,z) \in \Omega'$ and $Maj(x,y,z) \in \Omega'$, so that if the $H^{(i-1)}$ and $M^{(i)}$ inputs to the compression function sha'-256 of SHA'-256 both consist of Ω' words, then the resulting $H^{(i)}$ output also consists of Ω' words. Consequently:

- the complexity of a collision search on the restriction of the sha'-256 compression function to input values belonging to Ω' is only 2^{64} instead of 2^{128} (since for such values a collision on the left half of each output word implies a collision on the whole output word);
- the complexity of a collision search on the SHA'-256 hash function is also only 2^{64} instead of 2^{128}: to construct such a collision, one can for instance first search two 512-bit initial message blocks M_1 and $M'_1 \in \Omega'^{16}$ such that $sha' - 256(H_0, M_1) = sha' - 256(H_0, M'_1)$. The complexity for this sha'-256 collision search is 2^{64}. Now given any binary suffix message of any length $M_2 \in \{0,1\}*$, the $M = M_1 \| M_2$ and $M' = M'_1 \| M_2$ messages provide a collision for the SHA-256 hash function.

The above attack can easily be generalized to the SHA"-256; SHA"'-256, etc. variants of SHA-256 in which all constants are selected in the subsets $\Omega'' = \{C \in \{0,1\}^{32} \mid \exists c \in \{0,1\}^8, C = c\|c\|c\|c\}$, $\Omega''' = \{C \in \{0,1\}^{32} \mid \exists c \in \{0,1\}^4, C = c\|c\|c\|c\|c\|c\|c\|c\}$, etc., of Ω instead of Ω'. This results in collision attacks of complexity only $2^{256/4} = 2^{64}$ for SHA'-256, $2^{256/8} = 2^{32}$ for SHA"-256 and so on.

We have checked these results experimentally by implementing the case of SHA"'-256. In other words, for SHA-256, we applied the described modifications to the compression function such that all constants were replaced by 32-bit values showing 8 identical nibbles. Next we generated a large hash table and searched for collisions on the compression function for 2^{20} input messages of length one block (512 bits) showing 8 identical nibbles in each one of the 16 32-bit input words. On average, the number of collisions we expect to obtain is $\frac{2^{20} \times 2^{20}}{2 \times 2^{32}}$ which is about 2^7. These numbers were confirmed by our experiments.

As an illustration, in table 3 we provide two messages showing the required symmetry, a set of initial vectors showing the required symmetry, and the resulting collision we obtained on the compression function of SHA"'-256. (For this example, the 64 32-bit constants K_t were obtained by repeating 8 times their respective first nibble.)

Similarly, it is easy to define variants SHA'-512/384, SHA"-512/384, etc., of SHA-512 in which constants are replaced by more symmetric values, and to show that the collision search complexities for these variants are only $2^{512/4} = 2^{128}$, $2^{512/8} = 2^{64}$, 2^{32}, etc.

Thus with some very simple modifications, one obtains variants of the hash functions SHA-256 and SHA-384/512 which are surprisingly weak. We note that similar modifications on SHA-1 and on SHA-0 would have the same effect (and would not need any change in the message schedule). However, this does by no means imply that the original hash functions are insecure, not even remotely, but it casts some doubts on these designs.

6 Conclusion

We have investigated the security of SHA-2. We have shown that neither Dobbertin's nor Chabaud and Joux's attacks on MD-type hash functions seem to

Table 3. Example of two symmetric message blocks compressing to the same symmetric value

	Chaining variables

Initialisation vectors	`0xaaaaaaaa 0xbbbbbbbb 0xcccccccc 0xdddddddd`
	`0xeeeeeeee 0xffffffff 0x00000000 0x11111111`

Message 1	`0x99999999 0xbbbbbbbb 0xffffffff 0x44444444`
	`0x99999999 0x00000000 0x00000000 0x00000000`
	`0x00000000 0x00000000 0x00000000 0x00000000`
	`0x00000000 0x00000000 0x00000000 0x00000000`

Output 1	`0x00000000 0xbbbbbbbb 0x77777777 0xeeeeeeee`. `0x99999999 0x99999999 0x11111111 0x88888888`

Message 2	`0xeeeeeeee 0x00000000 0xffffffff 0x33333333`
	`0xffffffff 0x00000000 0x00000000 0x00000000`
	`0x00000000 0x00000000 0x00000000 0x00000000`
	`0x00000000 0x00000000 0x00000000 0x00000000`

Output 2	`0x00000000 0xbbbbbbbb 0x77777777 0xeeeeeeee`
	`0x99999999 0x99999999 0x11111111 0x88888888`

apply to SHA-2. Most features of the basic components of SHA-2 seem to provide a better security level than for preceding hash functions, even though the relative number of rounds is somewhat lower than for SHA-1 for instance, and though the selection criteria and security arguments for some design choices are difficult to reconstruct from the specification, in the absence of any public design report. Finally, we have shown that a simplified version of SHA-2 where the round constants are symmetric and where addition is replaced by exclusive or, is insecure.

Acknowledgements

This work is based on the result of an evaluation requested by the Japanese CRYPTREC project: http://www.ipa.go.jp/security/enc/CRYPTREC/... .../index-e.html. We would like to thank the members of the CRYPTREC project for having authorized this publication. We also thank the anonymous reviewers for their suggestions and comments.

References

[1] E. Biham, O. Dunkelmann, N. Keller, *Rectangle Attacks on 49-Round SHACAL-1*, in FSE 2003, Pre-proceedings of the conference, pages 39-48, 2003.

[2] J. Black, P. Rogaway, T. Shrimpton, *Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV*, in Advances in Cryptology - Crypto'02, pages 320-335, LNCS 2442, Springer-Verlag, 2002.

[3] B. den Boer and A. Bosselaers, *An attack on the last two rounds of MD4*, in Advances in Cryptology - Crypto'91, pages 194-203, LNCS 576, Springer-Verlag, 1992. 175

[4] F. Chabaud and A. Joux, *Differential Collisions in SHA-0*, in Advances in Cryptology - CRYPTO'98, LNCS 1462, pages 56-71, Springer-Verlag, 1998. 176

[5] I. B. Damgård, *A design principle for hash functions*, in Advances in Cryptology - Crypto'89, LNCS 435, pages 416-427, Springer Verlag, 1990. 177, 184

[6] C. Debaert, H. Gilbert, *The RIPEMD^L and RIPEMD^R Improved Variants of MD4 are not Collision Free,* in Fast Software Encryption - FSE'2001, LNCS 2355, Springer Verlag, 2001. 176

[7] H. Dobbertin, *Cryptanalysis of MD4*, in Journal of Cryptology vol.11 n.4, Springer-Verlag, 1998. 175, 184

[8] H. Dobbertin, *Cryptanalysis of MD5 Compress*, Presented at the rump session of Eurocrypt '96, May 14, 1996. 175, 184

[9] H. Dobbertin, *The status of MD5 after a recent attack*, CryptoBytes, vol. 2, n. 2, 1996. 175, 184

[10] H. Dobbertin, *RIPEMD with two round compress function is not collision-free*, in Journal of Cryptology vol.10 n.1, Springer-Verlag, 1997. 176, 184

[11] H. Dobbertin, A. Bosselaers and B. Preneel, *RIPEMD-160: a strengthened version of RIPEMD*, April 1996. http.esat.kuleuven.ac.be/pub/COSIC/bossselae/ripemd. 175

[12] H. Handschuh, L. Knudsen, M. Robshaw, *Analysis of SHA-1 in encryption mode,* in Topics in Cryptology - CT-RSA 2001, LNCS 2020, pages 70-83, Springer-Verlag, 2001. 179, 180, 185

[13] H. Handschuh, D. Naccache, *SHACAL: A Family of Block Ciphers* Submission to the NESSIE project, 2002. Available from `http://www.crytponessie.org` 185

[14] National Institute of Standards and Technology (NIST) *FIPS Publication 180-1: secure Hash Standard*, April 1994. 176

[15] National Institute of Standards and Technology (NIST), FIPS 180-2, 2002. `http://csrc.nist.gov/encryption/tkhash.html`. 176, 177

[16] National Institute of Standards and Technology (NIST) *FIPS Publication 197: Advanced Encryption Standard (AES)*, 2001. 176

[17] P. C. van Oorschot, M. J. Wiener, *Parallel Collision Search with Cryptanalytic Applications*, in Journal of Cryptology vol.12 n.1, Springer-Verlag, 1999.

[18] B. Preneel, R. Govaerts, J. Vandewalle, *Differential cryptanalysis of hash functions based on block ciphers*, Proc. 1st ACM Conference on Computer and Communications Security, pages 183-188, 1993. 185

[19] *RIPE Integrity Primitives for Secure Information Systems - Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040)*, LNCS 1007, Springer-Verlag, 1995. 175

[20] R. L. Rivest, *The MD4 message digest algorithm*, in Advances in Cryptology - Crypto'90, pages 303-311, Springer-Verlag, 1991. 175

[21] R. L. Rivest, *RFC1321: The MD5 message digest algorithm*, M. I. T. Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

[22] M.-J. O. Saarinen, *Cryptanalysis of Block Ciphers Based on SHA-1 and MD5*, in FSE'2003, Pre-proceedings of the conference, pages 39-48, 2003. 176, 179, 185

[23] S. Vaudenay, *On the need for multipermutations: Cryptanalysis of MD4 and SAFER*, in Fast Software Encryption - FSE'95, LNCS 1008, pages 286-297, Springer-Verlag, 1995. 175

A Example of 9-Round Differential Collision Patterns

SHA-384/512 Case

The following values are an example of the consecutive contents of the differences in the SHA-384/512 registers a, b, c, d, e, f, g, h when a “perturbative pattern” of Hamming weight one followed by the corresponding “corrective pattern” is applied to nine consecutive message words W.

The values of the 9 consecutive differences in words W[0] to W[8] are the following :

W[i]: 0x20 0x 0	W[i+1]: 0x50000000 0x880208
W[i+2]: 0x8a31001 0x4200000	W[i+3]: 0x 0 0x 0
W[i+4]: 0x20 0x 0	W[i+5]: 0x50000000 0x880208
W[i+6]: 0x 0 0x 0	W[i+7]: 0x 0 0x 0
W[i+8]: 0x20 0x 0	W[i+9]: 0x 0 0x 0

The corresponding values of the 10 consecutive differences in registers a, b, c, d, e, f, g, h (represented by 2 32-bit half registers separated by a .) are the following:
Round i : 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0
Round i+1 : 0x20 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x20 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0
Round i+2 : 0x 0 0x 0 . 0x20 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x40000000 0x208 . 0x20 0x 0 . 0x 0 0x 0 . 0x 0 0x 0
Round i+3 : 0x 0 0x 0 . 0x 0 0x 0 . 0x20 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x40000000 0x208 . 0x20 0x 0 . 0x 0 0x 0
Round i+4 : 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x20 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x40000000 0x208 . 0x20 0x 0
Round i+5 : 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x20 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x40000000 0x208
Round i+6 : 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x20 0x 0 . 0x 0 0x 0 . 0x 0 0x 0
Round i+7 : 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x20 0x 0 . 0x 0 0x 0
Round i+8 : 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x20 0x 0
Round i+9 : 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0 . 0x 0 0x 0

SHA-256 Case

The following values are an example of the consecutive contents of the differences in the SHA-256 registers a, b, c, d, e, f, g, h when a ” perturbative pattern ” (1-bit difference on the least significant bit) followed by a ” corrective pattern ” is applied to nine consecutive message words W.

Round i : 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0
Round i+1 : 0x 1 0x 0 0x 0 0x 0 0x 1 0x 0 0x 0 0x 0
Round i+2 : 0x 0 0x 1 0x 0 0x 0 0x40080400 0x 1 0x 0 0x 0
Round i+3 : 0x 0 0x 0 0x 1 0x 0 0x 0 0x40080400 0x 1 0x 0
Round i+4 : 0x 0 0x 0 0x 0 0x 1 0x 0 0x 0 0x40080400 0x 1
Round i+5 : 0x 0 0x 0 0x 0 0x 0 0x 1 0x 0 0x 0 0x40080400
Round i+6 : 0x 0 0x 0 0x 0 0x 0 0x 0 0x 1 0x 0 0x 0
Round i+7 : 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 1 0x 0
Round i+8 : 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 1
Round i+9 : 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0 0x 0

B Investigation of Differential Characteristics for SHA-2

Search for Low Weight Differential Characteristics over a Few Rounds:

The following table shows the evolution of the lowest weight differential characteristic over 4 rounds we found. Only the weight of the difference in each register is shown, as a circular rotation of the corresponding indexes of the initial difference bits achieves the same overall propagation pattern. We stress that in this setting, all the message words are identical: Only the input registers differ in a few bits.

probability	a	b	c	d	e	f	g	h
	0	0	0	0	1	0	0	3
1/16	0	0	0	0	0	1	0	0
1/2	0	0	0	0	0	0	1	0
1/2	0	0	0	0	0	0	0	1
1/4	1	0	0	0	1	0	0	0

- The first difference bit in register E affects both the *Choice* and the non-linear Σ_1 function. With probability $\frac{1}{2}$, the output of the Σ_1 function is equal to zero, and with probability 2^{-3} the 3 input difference bits in register H added to the 3 difference bits generated by the Σ_1 function do not generate any difference carry bits. Thus after the first round, with probability 2^{-4} only one difference bit propagates in register F.
- In rounds 2 and 3, with probability $\frac{1}{2}$, this difference bit does not cause any difference in the output of the *Choice* function. Thus we now have a one-bit difference in register H.
- In the last round of the characteristic, this one-bit difference is added respectively into register A and E, thus with probability $\frac{1}{4}$,the output difference is a 2-bit difference after 4 rounds.

Putting everything together, this low weight differential characteristic has a probability of 2^{-8}.

Search for Iterative Differential Characteristics

In order to investigate iterative differential characteristics for SHA-2 we approximated the actual differential transitions associated with each round of the SHA-2 register update function by a linear function of $\{0,1\}^{256}$ in the case of SHA-256 (resp. $\{0,1\}^{512}$ in the case of SHA-384/512) , by making the simplifying assumption that the output difference ($\delta a'$, $\delta b'$, $\delta c'$, $\delta d'$, $\delta e'$, $\delta f'$, $\delta g'$, $\delta h'$) associated with an input difference (δa, δb, δc, δd, δe, δf, δg, δh) is equal to the output difference one would obtain if SHA-2 the functions *Choice* and *Majority* were ignored and if '+' was replaced by '⊕'. Let us denote by L the 256 × 256 (resp. 512 × 512) binary matrix over GF(2) representing the above linear mapping. Let us denote by A and E the matrices associated with Σ_0 and Σ_1 respectively and by I and O the identity and the null 32 × 32 (resp. 64 × 64) matrices. L can be described as the following 8x8 block matrix:

$$L = \begin{pmatrix} A & O & O & O & E & O & O & I \\ I & O & O & O & O & O & O & O \\ O & I & O & O & O & O & O & O \\ O & O & I & O & O & O & O & O \\ O & O & O & I & E & O & O & I \\ O & O & O & O & I & O & O & O \\ O & O & O & O & O & I & O & O \\ O & O & O & O & O & O & I & O \end{pmatrix}$$

We are then using the matrix L to search candidate iterative differential characteristics over a restricted number of rounds r. For that purpose, we computed for each value r in $\{1, 16\}$ a basis of the kernel K_r of $L^r - I$,where I represents the 256 × 256 (resp. 512 × 512) identity matrix, using standard Gaussian reduction. Elements of K_r represent those input differences $\delta = (\delta a, \delta b, \delta c, \delta d, \delta e, \delta f, \delta g, \delta h)$ which stay invariant under r rounds, up to the approximation of the differential transition at each round by the linear function L. In other words, elements of K_r represent iterative characteristics for r rounds, provided the probabilities obtained when taking into account the approximations made in L are not too low.

In the case of SHA-256, we obtained for K_1 to K_{16} vector spaces of respective dimensions given by the following table.

$r =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$dim(K_r) =)$	1	2	1	4	1	2	4	8	1	2	1	4	1	8	1	16

To develop on particular example more in detail, K_4 has dimension 4. If we denote by a^n the concatenation of n binary words equal to a (so that for instance $(0111)^2 = 01110111$, etc.), a basis $\{e_{40}, e_{41}, e_{42}, e_{43}\}$ of K_4 is given by:

$$e_{40} = ((10)^{16}, 0^{32}, (01)^{16}, 0^{32}, 0^{32}, (10)^{16}, 0^{32}, (01)^{16})$$

$$e_{41} = ((01)^{16}, 0^{32}, (10)^{16}, 0^{32}, 0^{32}, (01)^{16}, 0^{32}, (10)^{16})$$

$$e_{42} = (0^{32}, (01)^{16}, 0^{32}, (10)^{16}, (01)^{16}, 0^{32}, (10)^{16}, 0^{32})$$

$$e_{43} = (0^{32}, (10)^{16}, 0^{32}, (01)^{16}, (10)^{16}, 0^{32}, (01)^{16}, 0^{32})$$

In the case of SHA-384/512, we obtained for K_1 to K_{16} vector spaces which dimensions given by the following table.

$r =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$dim(K_r =)$	2	4	2	8	2	4	8	16	2	4	2	8	2	16	2	32

As could be seen in the enumeration of the K_r base vectors for the first values of r, K_r elements are highly symmetric, i.e. they consist of differences $\delta = (\delta a, \cdots, \delta h)$ such that the 32-bit or 64-bit patterns δa to δh be periodic, of period 32 or 16 or 8 for SHA-256 (resp. 64, 32, 16 or 8 for SHA-512). Thus, any non zero element of the first sets K_r contains at least some non zero periodic words and thus cannot have an extremely low weight. Therefore we think that the approach described above does not provide high probability iterative differentials for SHA-2.