Evaluation Report

Security Level of Cryptography – SHA-256

Dr. Helena Handschuh
Gemplus R&D
Security Technologies
34, rue Guynemer
F-92447 Issy-les-Moulineaux
FRANCE

Dr. Henri Gilbert
France Télécom R&D
DTL/SSR
38-40 rue du Général Leclerc
F-92794 Issy-les-Moulineaux Cedex 9
FRANCE

Issy-les-Moulineaux
31 January 2002.

TABLE OF CONTENTS

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.

More in detail, a strong cryptographic hash function *h* is usually expected to satisfy the following requirements :

(1) **Collision resistance :** it must be computationally infeasible to find any two distinct messages *M* and *M'* such that $h(M) = h(M')$. The best collision resistance one can hope to achieve with an *m*-bit hash function is upper bounded by the $O(2^{m/2})$ complexity of a birthday attack [Oorschot] .

(2) **Preimage resistance (one wayness) :** given the $h(M)$ hash value of an unknown message *M*, it must be computationally infeasible to find any message *M'* (equal or not to *M*) such that $h(M') = h(M)$. The best preimage resistance one can hope to achieve with an *m*-bit hash function is upper bounded by the $O(2^m)$ complexity of an "exhaustive" search.

(3) **Second preimage resistance (weak collision-resistance) :** given any *M* known message and its $h(M)$ hash value, it must be computationally infeasible to find any *M*' message distinct from *M* such that $h(M') = h(M)$. The best preimage resistance one can hope to achieve with an *m*-bit hash function is upper bounded by the $O(2^m)$ complexity of an "exhaustive" search.

Moreover, no better collision search, preimage search or second preimage search attack than the generic attacks mentioned in the definition of requirements (1) (2) and (3) must be known for *h*.

Requirement (1) is by far the most important one in practice for the assessment of any candidate hash function such that the one considered in this report, since :

- A collision resistant hash function is necessarily second preimage resistant,i.e. (1) $\Rightarrow$(3) ;
- Although some artificial counter examples of the implication (1) $\Rightarrow$ (2) are easy to construct, it can be conjectured that for practical hash function candidates such as the one considered in this report, any computationally feasible preimage search attack would automatically result in a computationally feasible collision search attack. As a matter of fact, to search for a collision assuming a preimage search attack would exist, one just would have to draw a sufficient number of M message from a sufficiently large set of messages, and then to apply the preimage computation attack to h(M) until eventually an M' preimage distinct from M would be found.

Thus, in order to assess the security of a candidate cryptographic hash function such as the one analysed in this report, it is nearly sufficient to restrict oneself to the investigation of the collision resistance properties of the considered function - and of the collision resistance properties on the underlying compression function (see Merkle Damgard construction hereafter).

Cryptographic hash functions represent a quite useful primitive in IT security. Depending on the nature of the application, they may provide :

- message integrity when used in a pre-processing step for digital signature algorithms;
- information integrity when attached to a(n) (encrypted) message;
- redundancy when appended to data before encryption;
- protection of passwords if preimage resistance is achieved;
- commitment in zero-knowledge authentication schemes ;
- derivation of a signature scheme from some zero knowledge schemes ;
- pseudo-random string generation or key derivation for general applications;
- a basis for constructions of MACs, stream ciphers and block ciphers.

Depending on the considered application, all or only part of the collision resistance, preimage resistance and second preimage resistance properties defined above may be required.

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. First padding data (which include filling bits and sometimes information such as the message length) is appended to the M message to be hashed as to obtain a padded message which length is a multiple of n, and then split into n-bit blocks M_1 to M_k. Denote the compression function by f and the hash function by h. The m-bit hash value $h(M)$ is computed using the recurrence $H_0=IV_0$ (where IV_0 is an m-bit constant initial value); for i=1 to k $H_i = f(H_{i-1}) ||M_i$; $h(M) = H_k$. We are using in the sequel a terminology introduced by *H.* Dobbertin to distinguish two kinds of situations where two distinct (IV,M) and (IV',M') inputs have the same image by the compression function of an iterated hash function: we will restrict the use of the term collision to the case where in addition $IV=IV'$ and use the term pseudo collision otherwise. It was independently shown by Merkle and Damgard [Damgard] that if a compression function f is collision and pseudo collision resistant, then under some simple additional conditions on the padding rule, the associated hash function is collision resistant.

The most commonly used and (for the most recent ones) most trusted existing cryptographic hash functions do all belong to the *MD*-family of iterated hash functions, which includes MD4 (a 128-bit hash function proposed in 1990 by R. Rivest [RivestMD4]), MD5 (a more conservative 128-bit hash function proposed in 1991 by R. Rivest [RivestMD5]), RIPEMD (a 128-bit function which mixes the outputs of two parallel improved variants of MD4, proposed in 1995 by the European RIPE consortium [RIPE]), RIPEMD-128 and RIPEMD-160 (128-bit and 160-bit variants of RIPEMD [RIPEMD160]), SHA (a 160-bit hash function proposed in 1992 by NIST), and SHA-1 (which was proposed by NIST as an improvement and replacement to the initial SHA [SHA1] in 1994). Finally, variant hash functions built around similar design principles than SHA-1, but characterized by a larger hash length value m have been recently proposed by NIST namely SHA-256 (m = 256), SHA-384 (m = 384), SHA-512 (m = 512). As far as we know, the main motivation for introducing new standard hash functions for this range of m values was to

provide hash functions which $2^{m/2} = 2^{128}$, 2^{196} and 2^{256} security levels against collision search attacks be consistent with the $2^{k} = 2^{128}$, 2^{196} and 2^{256} security levels corresponding the three standard key sizes k = 128, 196 and 256 of the recently adopted Advanced Encryption Standard.
The hash functions of the MD family and the associated compression functions have been submitted to an extensive cryptanalytic investigation during the past years. First, some collisions attacks were discovered on two of the three rounds of the MD4 compression function: an attack on the two last rounds, by den Boer and Bosselaers [BoerMD4], and an attack on the two first rounds leading to almost collisions on the full MD4, by Vaudenay [Vaudenay]. This provided initial arguments in favour of moving from MD4 to MD5. Later on, Dobbertin established three major results on the cryptanalysis of the MD family of hash functions, namely (1) collisions for the full MD4 compression and hash functions that led to recommending the abandonment of its use, (2) collisions for both the first and the last two rounds of the compression function of RIPEMD, and (3) pseudo collisions on the whole MD5 compression [DobMD5], which do not lead to a collision of the associated MD5 hash function, but nevertheless invalidate the applicability of the Merkle-Damgard construction. Finally, Chabaud and Joux [Chabaud] discovered an attack allowing to find collisions for SHA in approximately 2^{61} SHA computations (instead of the about 2^{80} computations an ideal 160-bit hash function would require). As a consequence of these analyses, MD4 and SHA are no longer recommended, RIPEMD-128 seems to be a more conservative 128-bit hash function than MD5 and RIPEMD, and RIPEMD-160 and SHA-1 seem to be far from reach of known attack methods.

The rest of this report is organised as follows. Section 2 briefly describes SHA-256, and makes some preliminary remarks on its main design feature and their comparison with the corresponding features of SHA-1. Section 3 investigates the applicability of the main currently known attacks of cryptographic hash functions to SHA-256. Section 4 investigates other potential attack directions. Finally, Section 5 concludes our report .

2 Description and first remarks

2.1 Outline of SHA-256

SHA-256 is an MD-type hash function and works as follows : first the message is right-padded with a binary ‘1’ then it is cut into blocks of 512 bits. If the length of the last block does not exceed 448 bits, as many zeros as necessary are appended to fill 448 bits and the binary length of the original message (before padding) is appended in the last 64 bits of the block to form a 512bit block. Else, the block is filled with zeros up to a length of 512 bits, and an extra block is appended filled with 448 zeros; again the binary length of the original message is appended in the last 64 bits to form a complete 512 bit block. This form of padding is non-ambiguous and is an example of a valid Merkle-Damgard strengthening.

After the padding phase, registers a, b, c, d, e, f, g, h are initialised to 8 pre-determined 32-bit constants H_0 to H_7 [SHA2] for the first message block, and to the intermediate hash value for the following blocks. Next, 64 rounds of the compression function are applied following the pseudo-code given hereunder.

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

Where the *Ch, Maj,* $\mathit{\Sigma_0}$, and $\mathit{\Sigma_1}$ functions are independent of the round number[1], and where K_t and W_t are a constant and a message word which value depends upon the round number t [SHA2]. Finally, the output of the registers is added to the previous intermediate hash value to give the new intermediate hash value, according to the Davies-Meyer construction. See Figure 2 for an outline of one round of the compression function.

The ‘message schedule’ takes the original 512-bit message block as input and expands these 16 32-bit words into 64 words, 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_1 and σ_0 [2]are given in [SHA2]. See Figure 1.

After all consecutive 512-bit message blocks have been hashed, the last intermediate hash value is the final overall hash value.

[1] The *Ch*, *Maj*, $\mathit{\Sigma_0}$, and $\mathit{\Sigma_1}$ functions operate on 32-bit input words, and produce the 32-bit words given by
$\mathit{Ch}(X,Y,Z) = (X \wedge Y) \oplus (\neg X \wedge Z)$;
$\mathit{Maj}(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$;
$\mathit{\Sigma_0}(X)= \mathit{ROTR}^2(X) \oplus \mathit{ROTR}^{13}(X) \oplus \mathit{ROTR}^{22}(X)$
$\mathit{\Sigma_1}(X)= \mathit{ROTR}^6(X) \oplus \mathit{ROTR}^{11}(X) \oplus \mathit{ROTR}^{25}(X)$

[2] The σ_0 and σ_1 functions operate on 32-bit input words, and produce the 32-bit words given by
$\sigma_0\ (X)= \mathit{ROTR}^7(X) \oplus \mathit{ROTR}^{18}(X) \oplus \mathit{SHR}^3(X)$
$\sigma_1\ (X)= \mathit{ROTR}^{17}(X) \oplus \mathit{ROTR}^{19}(X) \oplus \mathit{SHR}^{10}(X)$

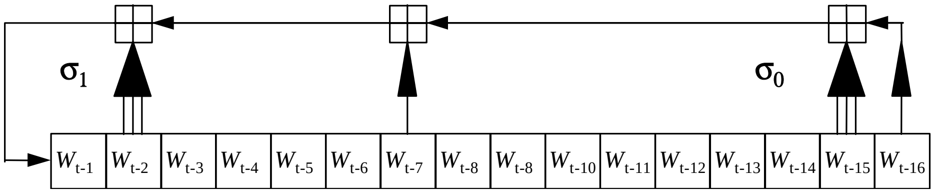

Figure 1 : (W_t) message schedule recurrence

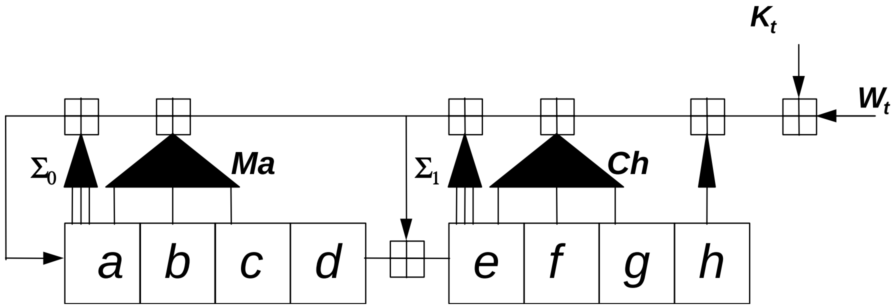

Figure 2 : Hash computation, state register update function

2.2 Preliminary remarks and comparison with SHA-1

2.2.1 Message schedule computations

The (W_t) message schedule computation is depicted in Figure 1 above. It can conveniently be represented by a 16-stages feedback register.

Its overall structure is similar to the one used in SHA-1 : in both cases (W_t) is produced by a recurring sequence of depth 16 initialised with the 16-words message block $M = M_0, M_1, \ldots, M_{15}$. The most significant differences between the SHA-1 and SHA-256 message schedules are the following :

☺ Unlike the SHA-1 message schedule computations, the SHA-256 message schedule computations are not GF(2)-linear, because of the involvement of the + addition instead of ⊕. This complicates the message schedule computations, but on the other hand it makes the properties of the message schedule recurrence more difficult to analyse, because the set of possible difference patterns is no longer a linear code.

☺ The SHA-1 property that unlike in SHA-0, the recurrence relation mixes the various bit positions is strengthened, due to the involvement of the bit rotations in σ_0 and σ_1 (which play a similar role in the SHA-256 recurrence to the $ROTL^1$ rotation in the SHA-1 recurrence) and also because of the diffusion effect introduced by the + addition in the SHA-256 recurrence.

☺ The circular rotation invariance properties of input words are not preserved in the SHA-256 message schedule computations (whereas they are preserved in the SHA-0 and SHA-1 computations) : this is due to the use of + instead of ⊕ and to the fact that σ_0 and σ_1 do not only involve *ROTL* circular rotations, but also shift operations (see Section 4.2 for details).

☹ The message schedule length to working variable register length ratio, which represents the number of "full rotations" of the working variable register during each compression function computation, is much lower in the case of SHA-256 as in the case of SHA-1 : 64/8 = 8 only instead of 80/5 = 16 for SHA-1. The exact performance to security balance argumentation behind the substantial diminution of this ratio are unclear to us. This may look at first glance as a serious decrease of the security margin offered by SHA-256. On the other hand one may probably consider that the decrease of this ratio is at least partly compensated by the higher complexity of the working variable update function of SHA-256 and the fact that unlike in SHA-1 two register values are substantially modified at each round.

2.2.2 State register update function

The overall structure of the 8 32-bit words SHA-256 register (*a,b,c,d,e,f,g,h*) update function performed at each round is quite similar to the one of the 5 32-bit words SHA-0 and SHA-1 register (a,b,c,d,e) update function. The following differences are however worth being noticed:

☺ The SHA-256 round function is substantially more complex than the SHA-0 and SHA-1 one and achieve stronger and faster diffusion effects. As a matter of fact : the Σ_0 and Σ_1 GF(2) linear functions achieve a faster diffusion and mixing of the various bit positions than the ROTL^5 an ROTL^{30} rotations of SHA-1 ; both the *Majority* and the *Choice* non-linear functions are applied at each round whereas at most one of these functions is applied in the case of SHA-1 ; two register words (namely these values contained in the d and h stages of the round input) are substantially modified at each round, whereas only the content of the e stage is substantially modified in the case SHA-1.

☹ The state register update function (see figure 2) is the same for all rounds (except of course from the facts that the K_t constants are pair-wise distinct and the W_t input words are generally distinct for distinct words). The SHA-0 and SHA-1 state register update functions are less uniform, since rounds 0 to 19, 20 to 39, 40 to 59 and 60 to 79 of these functions involve the *Choice*(*x*,*y*,*z*), *Xor*(*x*,*y*,*z*), *Majority*(*x*,*y*,*z*) and *Xor*(*x*,*y*,*z*) ternary functions respectively. It may be conjectured that this lesser uniformity represents a slight security advantage for SHA-1.

2.2.3 Majority and choice functions

In this section we investigate the elementary properties of the majority and choice functions as well as the modular addition operation.

- **Both the choice and majority functions operate on individual bits and are balanced on their respective input domains**.

Recall their respective formulae :

$$Ch(X,Y,Z) = (X \wedge Y) \oplus (\neg X \wedge Z) \ ;$$
$$Maj(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z) \ ;$$

We can compute the difference distribution table of their output with respect to a 3-bit input difference. The notation of the table 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.

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 subsequent section 3.2, it is useful to note that both functions achieve a zero output difference (i.e. an internal collision) with average probability ½.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 ½ 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.

  Thus on average, a one-bit difference before a modular addition does not propagate after the addition operation with probability ½.

2.2.4 Sigma functions

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

- **The Σ_0 and Σ_1 GF(2)- linear mappings are one to one**. This is a simple consequence of the fact that if one represents any 32-bit word $W = (W[0]W[1]....W[32])$ as a $GF(2)[X]/X^{32}+1$ polynomial $W[0] + W[1]\ X + W[2]\ X^2 + \ldots + W[31]\ X^{31}$ then Σ_0 and Σ_1 are represented by a multiplication by the $X^2+X^{13}+X^{22}$ and $X^6+X^{11}+X^{25}$ polynomials of $GF(2)[X]$ respectively, and of the observation that these two polynomials are co-prime with the $X^{32}+1 = (X+1)^{32}$ polynomial of $GF(2)[X]$.

- **The σ_0 and σ_1 GF(2)- linear mappings are one to one** (in order to check this property, we computed the 32x32 GF(2) matrices representing σ_0 and σ_1 , and checked that the kernel of this matrix was restricted to the null vector).

2.2.5 Constants

The choice of the $H_0^{(0)}$ to $H_7^{(0)}$ and K_0 to K_{63} constants plays an important role in the security of SHA-256 , since :

- the main difference between the various rounds of the SHA-256 compression function consists in the use of pair-wise distinct K_i constants (see Section 2.2.2 above) ;

- the use of too symmetric constant values would lead to strong weaknesses in close variants of SHA-256 (see Section 4.2 hereafter).

We believe the way the actual SHA-256 $H_0^{(0)}$ to $H_7^{(0)}$ and K_0 to K_{63} constant values are derived is appropriate to avoid any undesirable symmetry property. [They are determined by the fractional parts of the square roots (respectively the cube roots) of the first 8 (respectively 64) prime numbers].

3 Security of SHA-256 against known hash functions attack techniques

3.1 Investigation of the applicability of Dobbertin's attack techniques

The hash functions attack techniques introduced by H. Dobbertin in [DobMD4, DobMD5, DobRIPEMD] take opportunity of the extremely simple structure of the message schedule of hash functions such as MD4, MD5 and RIPEMD. In such functions, the 16-words of the message block are just repeated in permuted order a certain number r of times : r = 3 in the case of MD4 and RIPEMD, and r = 4 in the case of MD5. As a consequence, it is trivial with such functions to create (M,M*) pairs of message blocks producing an extremely low weight difference pattern at the output of the message schedule : if M and M* only differ in one bit of only one of their 16 words, then there are only r differences, of weight only 1 each, at the output of the message schedule.

Dobbertin's attacks are using such (M,M*) pairs of extremely low weight difference (e.g. weight 1). The attack strategy consists in controlling the diffusion of the r resulting message schedule output differences through the hash function computations, in order for the state registers differences caused by the r-1 message schedule differences encountered in the r-1 first 16-step rounds to be cancelled by the last message schedule difference encountered in the last 16-step round. Depending on the considered steps of the compression function, the control method may consist :

- in differential techniques (one simply expects certain input differences to result in certain output differences at the output of the state register update function)
- or in more sophisticated equations solving techniques, allowing for instance to fix all the register values encountered in a small number k of consecutive steps of the computation as to allow a simple message schedule input difference, e.g. of weight 1, at the first of the k steps, to result in a prescribed difference value after k steps.

Another example of the application of Dobbertin's techniques of [DobMD4, DobMD5, DobRIPEMD] to collision attacks can be found in [Debaert].

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-256, Dobbertin's attacks do not seem applicable to these functions. More explicitly, the recurrence relation of the SHA-256 function (in particular the $\sigma_0(W_{t-2})$ term in this recurrence) ensures a fast and strong diffusion of any low weight difference in the M message block, 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.

3.2 Investigation of the applicability of Chabaud and Joux's attack techniques

Unlike Dobbertin's attack 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 the GF(2) linearity of this expansion function- to construct relatively low weight differences on the W message schedule output[3] which are likely to produce collisions on the SHA-0 compression function.

Chabaud and Joux's attack can be roughly summarised as follows. One first identifies "corrective patterns", i.e. sets of W difference bits positions allowing to cancel after a few rounds, with a sufficiently high probability the differences introduced in the SHA-0 state register by the introduction, in one of the W words say W_t) of a "perturbative pattern" consisting of a one-bit difference. Then low weight sequences of 1-bit "perturbative patterns" satisfying the recurrence of the SHA-0 message schedule (and some extra easy technical conditions) are identified. Due to the structure of the SHA-0 message schedule, the ΔW difference pattern resulting from the

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

superposition of these various perturbative patterns and of all the corresponding corrective pattern automatically satisfies the linear recurrence of the message schedule. Therefore, numerous pairs of messages leading to the ΔW difference pattern 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-256 faster than by exhaustive search. Using the differential properties of the non-linear functions shown in section 2.2, we approximate each addition by an exclusive or operation with probability ½, and the majority and choice functions by zero with probability ½. We proceed in three steps :

- define a low weight perturbation and related corrective patterns
- compute the associated probabilities
- produce heuristic evidence that these patterns may **not** be generated according to the SHA-256 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 consecutive round, to disable the propagation into the A register by appropriate corrective patterns of the next message words. We believe 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 the following table :

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

Fact In other words, 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 basic differential collision pattern over 9 rounds.

An explicit example of such a difference propagation on the 8 registers is given in Annex A.

- The next step is to evaluate the probability of the preceding 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 D register to form the new E register value ;
  - The addition of T_1 and T_2 to form the new A register value.

  The probability of a carry bit appearing in one of these additions is bounded by ½ for each addition, thus we upper bound the overall probability for two additions per difference bit by ¼ . There are a total of 1+6+9+1+6+1=24 difference bits over the 9-round pattern. Thus 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 ½ 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 *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) so as 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. For obvious reasons, 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.

Looking at the probabilities, if two of these patterns are applied, the overall associated probability already becomes lower than the success probability of a birthday attack, which is 2^{-128} for SHA-256.

Fact *The overall probability associated to two 9-round differential collision patterns is upper bounded by* 2^{-132} *which is far lower than the natural birthday attack bound.*

In conclusion, The original attack by Chabaud and Joux on SHA-0 does not extend to SHA-256.

4 Investigation of other dedicated attacks

4.1 Differential attacks

4.1.1 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 is possible to find any pseudo-collisions on the compression function of SHA-256, then the Merkle-Damgard security argument cannot be applied. In other words, the existence of a pseudo-collision on the compression function of SHA-256 would represent an undesirable propertyFurthermore, if multiple partial pseudo-collisions could be found on the compression function, this would help constructing a collision on the overall hash function. As a matter of fact, if an exhaustive search is done over messages whose hash values already partially collide, the rest of the hash value collides with much higher probability than expected.

Thus it makes sense to investigate the differential properties of the compression function.

A special case for finding partial pseudo-collisions is to view the compression function as a block cipher encrypting registers A,B,C,D,E ,F,G, H under key [W_0 … W_{64}]. This has already been investigated for SHA-1 in [SHACAL]. No such partial collisions could be found. In the rest of Section 4.1, we investigate whether partial collisions of the same special form can be found for SHA-256.

4. 1. 2 Search for low weight differential characteristics over a few rounds

As in section 3.2, using the differential properties of the non-linear functions shown in section 2.2, we approximate each addition by an exclusive or operation with probability ½, and the majority and choice functions by zero with probability ½.

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
2^{-4}	0	0	0	0	0	1	0	0
½	0	0	0	0	0	0	1	0
½	0	0	0	0	0	0	0	1
¼	1	0	0	0	1	0	0	0

- The first difference bit in register E affects both the *Choice* and the non-linear $\sum_1$ function. With probability ½, the output of the $\sum_1$ 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 $\sum_1$ 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 ½, 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 ¼ ,the output difference is a 2-bit difference after 4 rounds.

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

For SHA-256, even though it does not concatenate over the whole 64 rounds, we can conclude that the best overall differential probability appears to be lower than $2^{-8*16} = 2^{-128}$ which is as low as the ‘natural’ collision probability on a 256-bit hash value. Thus a standard differential attack on the compression function is not anywhere near likely to succeed.

4. 1. 3 Search for iterative differential characteristics

In order to complement the search for high probability differential characteristics of the SHA-256 round register update function presented above, we investigated iterative differential differential characteristics of non negligible probability on a reduced number of rounds. For that purpose we used the following approach :

- We approximated the actual differential transitions associated with each round of the SHA-256 register update function by a linear function of $\{0,1\}^{256}$, by making the simplifying assumption that the (δa', δb', δc', δd', δe', δf', δg', δh') output difference associated with a (δa, δb, δc, δd, δe, δf, δg, δh) input difference is equal the output difference one would obtain if in the SHA-256 function the Choice and Majority functions were ignored and if + was replaced by ⊕.

- Let us denote by L the 256x256 binary matrix over GF(2) representing the above linear mapping. Let us denote by A and E the 32x32 matrices associated with $\mathbf{\Sigma_0}$, and $\mathbf{\Sigma_1}$ respectively, and by I and O the identity and the null 32x32 matrices. L can be described as a a 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 L matrix to identify candidate iterative differential characteristics over a restricted number r of rounds. For that purpose, we computed for each r value of [1,16] a basis of the K_r kernel of L^r-I_{256} ,where I_{256} is the 256x256 identity matrix, using standard Gaussian reduction (see annex B). The δ=(δa, δb, δc, δd, δe, δf, δg, δh) elements of K_r represent these elements δ=(δa, δb, δc, δd, δe, δf, δg, δh) 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, the K_r elements represent iterative characteristics for r rounds, provided the probabilities obtained when taking into account the fact that L represents only an approximation (not the actual transitions) are not too low.

As can be seen in the enumeration of the K_r base vectors for the first values of r listed in Annex B, K_r elements are highly symmetric, i.e. they consist of δ = (δa, δb, δc, δd, δe, δf, δg, δh) values such that the δa to δh 32-bit patterns be periodic, of period 32 or 16 or 8, and therefore, any non zero element of such K_r sets 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-256.

4.2 Weakness in SHA-256 variants with too symmetric constant values

In this Section we show that if some relatively slight variations are made in the SHA-256 specification, the resulting modified hash function does no longer offer any collision resistance. The considered variations essentially consisting in replacing all constants encountered in the algorithm by highly symmetric values.

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

- the $H_0^{(0)}$ to $H_7^{(0)}$ words of the $H^{(0)}$ initial hash values by 0^{32} or more generally by any 8 constant 32-bit words belonging to Ω ;
- the K_0 to K_{63} constants involved in the hash computation by 0^{32} or more generally by any 64 32-bit words belonging to Ω ;
- the + operation (mod 2^{32} addition) in the hash computation by $\oplus$;
- the $SHR^3(x)$ shift operation of the σ_0 function by the $ROTR^3(x)$ circular shift operation ;
- the $SHR^{10}(x)$ shift operation of the σ_1 function by the $ROTR^{10}(x)$ circular shift operation.

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 msha-256 compression function of MSHA-256 both consist of Ω words, then the resulting $H^{(i)}$ output also consists of Ω words. Consequently :

- The complexity of collision search on the restriction of the msha-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 collision search on the MSHA-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 msha-256(H_0,M_1) = msha-256(H_0,M'_1). The complexity for this msha-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 MSHA-256 hash function.

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

5 Conclusion

We have investigated several directions on the security of SHA-256. We have shown that neither Dobbertin's nor Chabaud and Joux's attacks on MD-type hash functions seem to transpose to

SHA-256. Most features of the basic components of SHA-256 provide a better security level than for preceding hash functions, even though the relative number of rounds may seem 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 mere specification, in the absence of any public design report. We have investigated differential properties of the underlying compression function and did not find any highly probable iterative characteristics, nor characteristics which extend to all rounds of the compression function. Finally, we have shown that a simplified version of SHA-256 where the round constants are half-wise symmetric is not secure.

In light of these observations, we conclude that none of the currently known attack methods can be successfully applied to SHA-256, and that we are not aware of any attack allowing to reduce the complexity of preimage or second preimage computations on SHA-256 to substantially less than 2^{256} or the complexity for collision and pseudo-collision search on SHA-256 to substantially less than the 'natural' birthday collision bound which is $2^{(256)/2} = 2^{128}$.

6 References

[Chabaud] F.Chabaud and A.Joux, *Differential Collisions in SHA-0*, extended abstract, in CRYPTO'98, LNCS 1462, pp 56--71, 1998.

[BoerMD4] B.den Boer and A.Bosselaers, *An attack on the last two rounds of MD4*, in Advances in Cryptology - Crypto'91 pages 194-203 LCNS 576 Springer-Verlag 1992.

[Damgard] I.B. Damgard, *A design principle for hash functions*, Advances in Cryptology – Crypto'89, LNCS, vol. 435, pp.416-427, Springer Verlag 1990.

[Debaert] C. Debaert, H. Gilbert, *The RIPEMD^L and RIPEMD^R Improved Variants of MD4 are not Collision Free*, Fast Software Encryption'2001, Lecture Notes in Computer Science, Springer Verlag.

[DobMD4] H.Dobbertin, *Cryptanalysis of MD4*, in Journal of Cryptology vol.11 n.4 Autumn 1998.

[DobMD5] H.Dobbertin, *Cryptanalysis of MD5 Compress*, Presented at the rump session of Eurocrypt '96, May 14, 1996.

[DobMD5] H. Dobbertin, *The status of MD5 after a recent attack*, CryptoBytes, vol. 2, n. 2, 1996, pp. 1, 3-6.

[Oorschot] P.C. Oorschot, M.J. Wiener, *Parallel collision search with application to hash functions and discrete logarithms*, Pric. Of the 2nd ACM Conference on Computer and Communications Security, ACM, 1994, pp 210-218.

[RIPEMD2] H.Dobbertin, *RIPEMD with two round compress function is not collision-free*, in Journal of Cryptology vol.10 n.1, Winter 1997.

[RIPEMD160] H.Dobbertin, A.Bosselaers and B.Preneel, *RIPEMD-160: a strengthened version of RIPEMD*, April 1996.\\{\it ftp.esat.kuleuven.ac.be/pub/COSIC/bossselae/ripemd}.

[SHA1] National Institute of Standards and Technology (NIST) *FIPS Publication 180-1: secure Hash Standard*, April 1994.

[SHA2] National Institute of Standards and Technology (NIST), draft FIPS 180-2, http://csrc.nist.gov/encryption/tkhash.html.

[SHACAL] 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.

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

[RivestMD4] R.L.Rivest, *The MD4 message digest algorithm*, in Advances in Cryptology - Crypto'90 pages 303-311 Springer-Verlag 1991.

[RivestMD5] R.L.Rivest, *RFC1321: The MD5 message digest algorithm*, M.I.T. Laboratory for Computer Science and RSA Data Security, Inc., April 1992.

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

ANNEX A

The following values are an example of the consecutive contents of the differences in 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.

Register: A   B   C   D   E   F   G   H

```
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
```

ANNEX B

This annex describes vector bases for the K^r sets of candidate 256-bit difference values $\delta = (\delta a, \delta b, \delta c, \delta d, \delta e, \delta f, \delta g, \delta h)$ for iterative characteristics over r rounds, r being comprised between 1 and 8. It also provides the dimensions of the K^r sets for higher r values.

We are using the following notational convention : for any binary word a, we denote by a^n the concatenation of n binary words equal to a. Thus for instance $(0111)^2 = 01110111$, etc.

$K^1 = \langle e_1 \rangle$ where

- $e_1 = (1^{32}, 1^{32}, 1^{32}, 1^{32}, 1^{32}, 1^{32}, 1^{32}, 1^{32})$

$K^2 = \langle e_{20}, e_{21} \rangle$ where

- $e_{20} = (1^{32}, 0^{32}, 1^{32}, 0^{32}, 1^{32}, 0^{32}, 1^{32}, 0^{32})$
- $e_{21} = (0^{32}, 1^{32}, 0^{32}, 1^{32}, 0^{32}, 1^{32}, 0^{32}, 1^{32})$

$K^3 = \langle e_1 \rangle$

$K^4 = \langle e_{40}, e_{41}, e_{42}, e_{43} \rangle$ where

- $e_{40} = ((10)^{16}, 0^{32}, (01)^{16}, 0^{32}, 0^{32}, (10)^{16}, 0^{32}, (01)^{16})$
- $e_{41} = ((01)^{16}, 0^{32}, (10)^{16}, 0^{32}, 0^{32}, (01)^{16}, 0^{32}, (10)^{16})$
- $e_{42} = (0^{32}, (01)^{16}, 0^{32}, (10)^{16}, (01)^{16}, 0^{32}, (10)^{16}, 0^{32})$
- $e_{43} = (0^{32}, (10)^{16}, 0^{32}, (01)^{16}, (10)^{16}, 0^{32}, (01)^{16}, 0^{32})$

$K^5 = \langle e_1 \rangle$

$K^6 = \langle e_{20}, e_{21} \rangle$

$K^7 = \langle e_{70}, e_{71}, e_{72}, e_{73} \rangle$ where

- $e_{70} = (1^{32}, 1^{32}, 0^{32}, 0^{32}, 0^{32}, 0^{32}, 0^{32}, 1^{32})$
- $e_{71} = (1^{32}, 0^{32}, 0^{32}, 0^{32}, 0^{32}, 0^{32}, 1^{32}, 0^{32})$
- $e_{72} = (1^{32}, 1^{32}, 0^{32}, 1^{32}, 0^{32}, 1^{32}, 0^{32}, 0^{32})$
- $e_{73} = (0^{32}, 1^{32}, 1^{32}, 0^{32}, 1^{32}, 0^{32}, 0^{32}, 0^{32})$

$K^8 = \langle e_{80}, e_{81}, e_{82}, e_{83}, e_{84}, e_{85}, e_{86}, e_{87} \rangle$ where

- $e_{80} = ((0010)^8, (10)^{16}, (1011)^8, (10)^{16}, 1^{32}, (0111)^8, 0^{32}, (0001)^8)$
- $e_{81} = ((0100)^8, (01)^{16}, (0111)^8, (01)^{16}, 1^{32}, (1110)^8, 0^{32}, (0010)^8)$
- $e_{82} = ((1000)^8, (10)^{16}, (1110)^8, (10)^{16}, 1^{32}, (1101)^8, 0^{32}, (0100)^8)$
- $e_{83} = ((0001)^8, (01)^{16}, (1101)^8, (01)^{16}, 1^{32}, (1011)^8, 0^{32}, (1000)^8)$
- $e_{84} = ((1011)^8, (01)^{16}, (1000)^8, (10)^{16}, 1^{32}, (0111)^8, (0001)^8, 0^{32})$
- $e_{85} = ((0111)^8, (10)^{16}, (0001)^8, (01)^{16}, 1^{32}, (1110)^8, (0010)^8, 0^{32})$
- $e_{86} = ((1110)^8, (01)^{16}, (0010)^8, (10)^{16}, 1^{32}, (1101)^8, (0100)^8, 0^{32})$
- $e_{87} = ((1101)^8, (10)^{16}, (0100)^8, (01)^{16}, 1^{32}, (1011)^8, (1000)^8, 0^{32})$

The dimensions of the next eight vector-spaces are respectively 1,2,1,4,1,8,1,16.