SHACAL

(— Submission to NESSIE —)

[Published in *Proc. of First Open NESSIE Workshop*, Leuven, Belgium, November 13–14, 2000.]

Helena Handschuh and David Naccache

Gemplus Card International
34 rue Guynemer, F-92447 Issy-les-Moulineaux, France
{helena.handschuh, david.naccache}@gemplus.com

Submission Statement

This submission presents and analyses the block cipher SHACAL, as a submission to NESSIE. It is based on the hash standard SHA-1 used in encryption mode. We believe the main strength of this block cipher is its inheritance from the extensive analysis that has been made on the hash function itself. We state that no hidden weakness has been inserted in this block cipher, and we believe the design principles to be sound. To the best of our knowledge, SHACAL is not covered by any patents. We do not intend to apply for any patent covering SHACAL and undertake to update the NESSIE project whenever necessary.
The estimated computational efficiency of SHACAL is 2800 cycles per 20 byte block encryption, 2330 cycles per 20 byte block decryption and 3200 cycles per 64 byte key setup. Timing measurements are given for a PC using an amd K6 processor running at 233 Mhz. 1 million SHACAL encryptions take about 12 seconds and 1 million decryptions take about 10 seconds. 1 million key setups take 14 seconds.
The following report analyses the cryptographic hash function SHA in encryption mode. A detailed analysis is given of the resistance of SHACAL against the most powerful known attacks today. It is concluded that none of these attacks can be applied successfully in practice to SHACAL. Breaking SHA in encryption mode requires either an unrealistic amount of computation time and known/chosen texts, or a major breakthrough in cryptanalysis.
We would like to thank Lars R. Knudsen and Matt J. Robshaw for their extensive security analysis; without their help this submission would not have been possible.

1 Introduction

In the following we give a brief introduction to the Secure Hash Algorithm (SHA).

Many of the popular hash functions today are based on MD4 [5]. MD4 was built for fast software implementations on 32-bit machines and has an output of 128 bits. Because of Dobbertin's work [3, 2] it is no longer recommended to use

MD4 for secure hashing, as collisions has been found in about 2^{20} compression function computations.

In 1991 MD5 was introduced as a strengthened version of MD4. Other variants include RIPEMD-128, and RIPEMD-160. SHA was published as a FIPS standard in 1993. All these hash functions are based on the design principles of MD4. RIPEMD-128 produces hash values of 128 bits, RIPEMD-160 and SHA-1 produces hash values of 160 bits.

SHA was introduced by the American National Institute for Standards and Technology in 1993, and is known as SHA-0. In 1995 a minor change to SHA-0 was made, this variant known as SHA-1. The standard now includes only SHA-1. Descriptions of both algorithms follow.
Notation:

- $+$. Addition modulo 2^{32} of 32 bit words.
- $ROT_i(W)$. Rotate 32 bit word W to the left by i bit positions.
- $\oplus$. Bitwise exclusive-or.
- $\&$. Bitwise and.
- $|$. Bitwise or.

To hash a message proceed as follows.

1. Pad the message, such that the length is a multiple which fits the size of the compression function, see [4].
2. Initialize the chaining variables AA, BB, CC, DD, EE, each of 32 bits, by
   (a) $AA = IV_1 = \texttt{67452301}_x$,
   (b) $BB = IV_2 = \texttt{EFCDAB89}_x$,
   (c) $CC = IV_3 = \texttt{98BADCFE}_x$,
   (d) $DD = IV_4 = \texttt{10325476}_x$,
   (e) $EE = IV_5 = \texttt{C3D2E1F0}_x$.
3. For each message block of 512 bits:
   (a) Set $AA = A, BB = B, CC = C, DD = D, EE = E$.
   (b) Expand the 512 bits to 2560 bits, cf. later.
   (c) Compress the 2560 bits in a total of 80 steps; each step updates in turn one of the working variables A, B, C, D, and E, see section on compression function.
   (d) Set $AA = AA + A, BB = BB + B, CC = CC + C, DD = DD + D$ and $EE = EE + E$.
4. Output the hash value $[AA \parallel BB \parallel CC \parallel DD \parallel EE]$.

1.1 Compression function

Let the message blocks of 512 bits be denoted $M = [W^0 \parallel W^1 \parallel \ldots \parallel W^{15}]$, where W_i are 32-bit words. For SHA-0 the expansion of 512 bits to 2560 bits is defined

$$W^i = W^{i-3} \oplus W^{i-8} \oplus W^{i-14} \oplus W^{i-16}, \quad 16 \leq i \leq 79. \tag{1}$$

In SHA-1 the expansion is defined

$$W^i = ROT_1(W^{i-3} \oplus W^{i-8} \oplus W^{i-14} \oplus W^{i-16}), \quad 16 \leq i \leq 79. \tag{2}$$

These expansions are the only difference between SHA-0 and SHA-1.
Define the following functions.

$$f_{if}(X,Y,Z) = (X\&Y)|(\neg X\&Z) \tag{3}$$
$$f_{xor}(X,Y,Z) = (X \oplus Y \oplus Z) \tag{4}$$
$$f_{maj}(X,Y,Z) = ((X\&Y)|(X\&Z)|(Y\&Z) \tag{5}$$

The above 80 steps are then defined

$$A^{i+1} = W^i + ROT_5(A^i) + f^i(B^i, C^i, D^i) + E^i + K^i \tag{6}$$
$$B^{i+1} = A^i \tag{7}$$
$$C^{i+1} = ROT_{30}(B^i) \tag{8}$$
$$D^{i+1} = C^i \tag{9}$$
$$E^{i+1} = D^i \tag{10}$$

for $i = 0 \ldots, 79$, where

$$f^i = f_{if}, \quad 0 \le i \le 19$$
$$f^i = f_{xor}, \ 20 \le i \le 39, 60 \le i \le 79$$
$$f^i = f_{maj}, \ 40 \le i \le 59.$$

The constants K^i are defined

$$K^i = \texttt{5A827999}_x, \ 0 \le i \le 19$$
$$K^i = \texttt{6ED9EBA1}_x, \ 20 \le i \le 39$$
$$K^i = \texttt{8F1BBCDC}_x, \ 40 \le i \le 59$$
$$K^i = \texttt{CA62C1D6}_x, \ 60 \le i \le 79$$

The output after 80 steps, $A^{80}, B^{80}, C^{80}, D^{80}, E^{80}$ is then used to update the chaining variables AA, BB, CC, DD, EE.

In the following, Round 1 will refer to the first 20 steps, Round 2 to the next 20 steps and so on.

The best attack known on SHA-0 when used as a hash function is by Chabaud and Joux [1]. They show that in about 2^{61} evaluations of the compression function it is possible to find two messages hashing to the same value. A brute-force attack exploiting the birthday paradox would require about 2^{80} evaluations. There are no attacks reported on SHA-1 in the open literature. In the following we shall consider only SHA-1.

1.2 SHACAL or using SHA in encryption mode

SHA was never defined to be used for encryption. However, the compression function can be used for encryption. Each of the above 80 steps are invertible in the A, B, C, D, and E variables. Therefore, if one inserts a secret key in the message and a plaintext as the initial value, one gets an invertible function from

the compression function by ignoring the final addition with the initial values. This is the encryption mode of SHA considered in this report. Thus SHACAL is a 160-bit block cipher defined for a 512-bit secret key. Shorter keys may be used by padding the key with zeroes to a 512-bit string. However, SHACAL is not intended to be used with a key shorter than 128 bits.

2 Attacking SHA in encryption mode

The two best known attacks on systems similar to SHA in encryption mode are *linear cryptanalysis* and *differential cryptanalysis*. There has been a wide range of variants of the two attacks proposed in the literature but the basic principles are roughly the same. Also, many other attacks on encryption schemes have been suggested but they are less general than the two above mentioned ones, and do not apply to encryption schemes in general. In this report we shall consider only linear cryptanalysis and differential cryptanalysis. These attacks apply to SHA in encryption mode, but as we shall see, the complexities of attacks based on there approaches are completely impractical, if possible at all.

SHA uses a mix of two group operations, modular additions modulo 2^{32} and exclusive-or (bitwise addition modulo 2). If we use the binary representation of words, i.e., $A = a_{w-1}2^{w-1} + \cdots + a_1 2 + a_0$, and similarly for S, the binary representation of the sum $Z = A + S$ may be obtained by the formulae

$$z_j = a_j + s_j + \sigma_{j-1} \text{ and } \sigma_j = a_j s_j + a_j \sigma_{j-1} + s_j \sigma_{j-1}, \tag{11}$$

where σ_{j-1} denotes the carry bit and $\sigma_{-1} = 0$ (cf. [6]). This formulae will be used in the sequel several times.

2.1 Linear Cryptanalysis

Linear cryptanalysis attempts to identify a series of linear approximations A_i to the different operational components in a block cipher, be they S-boxes, integer addition, boolean operations or whatever. The individual linear approximations are then combined to provide an approximation for the greater proportion of the encryption routine. The combination of approximations is by simple bitwise exclusive-or so the final approximation is $A_1 \oplus A_2 \oplus \cdots \oplus A_n$.

If the linear approximations A_i hold with probability p_i then we define the bias to be $\epsilon_i = |p_i - 1/2|$. Provided $\epsilon_i \neq 0$ for each approximation A_i then they are potentially useful in a range of sophisticated linear cryptanalytic attacks. After combination, the overall bias of $A_1 \oplus A_2 \oplus \cdots \oplus A_n$ is typically estimated using the so-called *Piling-Up Lemma* as $\epsilon = 2^{n-1} \prod_{i=0}^{n-1} \epsilon_i$. If the final approximation over the bulk of SHA-1 has bias ϵ then the data requirements for an attack are given by $c \cdot \epsilon^{-2}$ where c is some constant that is dependent on the exact form of an attack. For the attacks we consider here practical experience suggests that $c \approx 8$, but to be conservative we will assume that $c = 1$.

We mentioned that we needed an approximation over the greater proportion of the cipher. Just as in differential cryptanalysis, there are a variety of tricks

and techniques available to the cryptanalyst to gain a few extra steps for free and they potentially allow the recovery of key material from the outer steps of the cipher at the same time. The number of outer steps that can be removed in this way is very specific to the approximations being used and the structure of the cipher. However we will see that the biases are so low with the linear cryptanalysis of SHA-1 that this level of detail is likely to be more little more than a distraction.

We will describe our approach. For each of the four rounds we will attempt to identify the longest perfect linear approximation in SHA-1 and what appear to be its most useful extensions. We will then make many conservative assumptions and use these approximations to assess a lower bound on the data requirements in a linear cryptanalytic attack.

Some Preliminaries In the analysis that follows we will typically only consider single-bit approximations across the different operations. Practical experience shows that attempts to use heavier linear approximations very soon run into trouble. While it is conceivable for some operations that heavier linear approximations will have a larger bias individually, it is usually much harder to use them as part of an attack and as such they are typically not useful. We will use the notation e_i to denote the single-bit mask used to form a linear approximation. Thus e_i is a 32-bit word that has zeros in all bit positions except for bit i. We will set the least significant bit position to be bit zero.

In all rounds there are four integer additions. However two of these are with constants; one is key material the other a round constant. At first it is tempting to ignore these two additions, but in fact the value of the key material has an important impact on the bias of the approximation.

Even without this consideration, using linear approximations across two (or more) successive additions is a complex problem. As an example, we might consider addition across two integer additions $x = (a + b) + c$. Consider the first integer addition $y = a + b$ in isolation. Then the bias for the linear approximations $a[i] \oplus b[i] = y[i]$ $(0 \leq i \leq 31)$ is $2^{-(i+1)}$. If we were then to consider the second integer addition $x = y + c$ we might be tempted to use the Piling-Up Lemma directly, but that would give us misleading results.

For example, in bit position $i = 2$, the Piling-Up Lemma would tell us that the approximation holds with bias $2^{-3} \times 2^{-3} \times 2 = 2^{-5}$. But note that the output from one integer addition is used directly as the input to the second integer addition thus this two operations are not independent. Instead, if we evaluate the boolean expressions directly using the least significant three bits of a, b, and c then we find that the bias is in fact 2^{-3}.

In the case of SHA-1 we have an even more complicated situation. We have the following string of additions that we need to approximate $x = (a+b)+k+c$ where k is a key- (and round-) dependent constant. The approximation we plan to use is $x[i] = a[i] + b[i] + k[i] + c[i]$ $(0 \leq i \leq 31)$. The bias that is observed will depend on the value of k.

Let us consider a simplified case, $x = k + y$. Imagine we make the approximation $x[i] = k[i] + y[i]$ $(0 \leq i \leq 31)$, where $y[i]$ is plaintext dependent bit and where $k[i]$ is a (fixed) key bit. Clearly if we consider only the least significant bit, $i = 0$, then the approximation always holds. For bit $i = 1$, the approximation holds always if $k[0] = 0$, but only with probability 0.5, that is bias zero, if $k[0] = 1$. If we are using bit $i \geq 1$ for the approximation then integers k for which $(k \;\&\; (2^i - 1)) = 0$ give a maximum bias, since there will be no carry bits in bit positions lower than i, and the approximation holds always, see formulae (11). This maximum is less than or equal to 2^{-2} for any bit position $i > 1$. Note that the number of these "weaker" keys that give a maximal bias is dependent on the bit position i. When $i = 2$ we have that one in four keys gives the maximal bias. If $i = 30$ then we have that only one key in 2^{30} gives this maximal bias. We also note that some values of k give a zero bias. Namely values of k that satisfy $(k \;\&\; (2^i - 1)) = 2^{i-1}$. For such values there are no carry bits for positions less than $i-1$. But since $k[i-1] = 1$ in this case, there will be a carry bit in position i if and only if $y[i-1] = 1$. If y is allowed to vary over all values (the approach usually taken in linear cryptanalysis) then the approximation $x[i] = k[i] + y[i]$ holds with probability 0.5, thus zero bias.

All rounds The cyclical structure of SHA-1 means that in all four rounds we can readily identify a family of linear approximations that always hold over four steps. We use Γ to denote a general pattern of bits to be used in the approximation and x^c to denote the left rotation of a 32-bit word x by c bit positions.

A	B	C	D	E	*bias*
Γ	-	-	-	-	
		$\downarrow$			1/2
-	Γ	-	-	-	
		$\downarrow$			1/2
-	-	Γ^{30}	-	-	
		$\downarrow$			1/2
-	-	-	Γ^{30}	-	
		$\downarrow$			1/2
-	-	-	-	Γ^{30}	

This is a "perfect" linear approximation over any four steps of SHA-1. In extending this approximation we will need to take into account the effects of the different boolean functions that are used in the different rounds. Our extended linear approximations will be formed according to these three rationale:

1. When approximating forward one step in any of the rounds, we try to avoid introducing an approximating bit in word E.
2. We try to use single-bit approximations in each word whenever possible, and we always try and use the least significant bit of a word.

3. We try and use as many internal cancellations as possible to keep the linear approximation as simple as possible.

These rationale do not necessarily guarantee that the linear approximations we construct are the best for the cryptanalyst. However they embody well-founded analytic techniques that are very likely to give the best constructable linear approximations.

Rounds 2 and 4 In these rounds the boolean function used to combine the words is the simple bitwise exclusive-or $\mathtt{b} \oplus \mathtt{c} \oplus \mathtt{d}$. This function in fact poses some difficulty to the cryptanalyst in terms of trying to manage the number of bits used in the approximations.

In Rounds 2 and 4 we can extend the basic "perfect" linear approximation that we have already shown for all rounds in the following way. This gives a linear approximation that acts over seven steps and holds with probability one (i.e. the bias is 1/2). In anticipation of its extension, we set $\Gamma = e_0$ according to our rationale in the previous section.

A	B	C	D	E	*bias*
e_2	-	-	-	-	
		$\downarrow$			1/2
-	e_2	-	-	-	
		$\downarrow$			1/2
-	-	e_0	-	-	
		$\downarrow$			1/2
-	-	-	e_0	-	
		$\downarrow$			1/2
-	-	-	-	e_0	
		$\downarrow$			1/2
e_0	e_{27}	e_{30}	e_0	e_0	
		$\downarrow$			1/2
e_0	$e_{27} \oplus e_0$	$e_{30} \oplus e_{25}$	$e_{30} \oplus e_0$	-	
		$\downarrow$			1/2
-	e_0	$e_{25} \oplus e_{30}$	$e_{25} \oplus e_{30}$	$e_{30} \oplus e_0$	

We conjecture that this is the longest "perfect" linear approximation over the steps in Rounds 2 and 4. If we are to use this in an attack then we will need to extend it. If we consider the only extension that is possible at the top then we have the following one-step linear approximation:

A	B	C	D	E
e_{29}	e_2	e_2	e_2	e_2
		$\downarrow$		
e_2	-	-	-	-

At the foot of the seven-step linear approximation we need to use the following one-step approximation:

A	B	C	D	E
-	e_0	$e_{25} \oplus e_{30}$	$e_{25} \oplus e_{30}$	$e_{30} \oplus e_0$
		$\downarrow$		
$e_{30} \oplus e_0$	$e_{27} \oplus e_{25}$	e_{28}	$e_{25} \oplus e_0$	$e_{25} \oplus e_0$

Using the techniques mentioned in the preliminary section, we estimate that the maximum bias for this nine-step linear approximation (taking into account the best possible value for the key material) is less than $2^{-2} \times 2^{-2} \times 2 = 2^{-3}$ and more than $2^{-3} \times 2^{-3} \times 2 = 2^{-5}$. This bias would apply to one in 2^{32} keys since we require a key condition on the approximation in step one and a key condition on the approximation in step nine. For roughly one in 2^2 keys there will be no bias to this linear approximation. The expected value of the bias might be expected to lie between $2^{-3} \times 2^{-3} \times 2 = 2^{-5}$ and $2^{-4} \times 2^{-4} \times 2 = 2^{-7}$. Experiments give that the bias using the best key conditions is around $2^{-4.0}$ and that the average bias over all keys is $2^{-5.6}$. For one in four keys there is no bias in the approximation.

We have identified a nine-step linear approximation. To facilitate our overall analysis we will add a step to this nine-step approximation. We could add a step at the beginning or at the end. It seems to be easier for the cryptanalyst to add the following one-step approximation to the beginning of the existing approximation.

A	B	C	D	E
$e_{24} \oplus e_2$	$e_{29} \oplus e_4$	$e_{29} \oplus e_2$	$e_{29} \oplus e_2$	e_{29}
		$\downarrow$		
e_{29}	e_2	e_2	e_2	e_2

Following our previous methods we will estimate that that maximum bias (under the most propitious key conditions for the analyst) lies in the range $(2^{-4}, 2^{-7})$ and that the average bias lies in the range $(2^{-7}, 2^{-10})$. For a little over one in four keys there will be no bias. Experiments demonstrate that the best key values (which occur for one in $2^{29+30+2}$ random keys) give a bias of $2^{-5.4}$ but that the bias for the average key is performing a little better than expected with a bias of $2^{-6.7}$. Since the case of the best key values is so rare, we propose to use 2^{-6} as a conservative representative of the bias of this ten-step linear approximation in Rounds 2 and 4.

Round 1 As in our analysis of Rounds 2 and 4 we consider the best extension to the basic four-step "perfect" approximation that applies in all rounds. Here the boolean function is $\mathtt{bc} \oplus (\mathtt{1} \oplus \mathtt{b})\mathtt{d}$. There are no perfect approximations across this operation, though there are several approximations with bias 2^{-2}.

Immediately we can see the following four-step extension to the existing basic linear approximation:

A	B	C	D	E	
-	-	-	-	e_0	
		$\downarrow$			1/4
e_0	e_{27}	-	e_0	-	
		$\downarrow$			1/2
-	e_0	e_{25}	-	e_0	
		$\downarrow$			1/4
e_0	e_{27}	-	$e_{25} \oplus e_0$	-	
		$\downarrow$			1/2
-	e_0	e_{25}	-	$e_{25} \oplus e_0$	

The bias for this extension can be computed as 2^{-3}. In extending further we need to approximate across the addition operation in a bit position other than the least significant. We will consider a worst-case scenario for the key values so that the bias of this approximation is perhaps around 2^{-2}. Of course it can be expected to be much less.

The following two-step extension allows us to form a ten-step approximation to the steps in Round 1 that holds with a bias of no more than 2^{-6} in the best case and in the range $(2^{-7}, 2^{-8})$ on average.

A	B	C	D	E
-	e_0	e_{25}	-	$e_{25} \oplus e_0$
		$\downarrow$		
$e_{25} \oplus e_0$	$e_{27} \oplus e_{20}$	-	e_0	-
		$\downarrow$		
-	$e_{25} \oplus e_0$	$e_{25} \oplus e_{18}$	-	e_0

Experiments confirm the ten-step linear approximation. The average bias was $2^{-7.2}$ and with the best key conditions (which hold for one in 2^{25} random keys) the bias over twenty trials was $2^{-6.4}$.

We will conservatively use 2^{-6} as the estimate for the bias for this ten-step linear approximation to the steps in Round 1.

Round 3 Once again we consider extensions to the basic linear approximation that applies in all rounds. Here the boolean function is $\texttt{bc} \oplus \texttt{cd} \oplus \texttt{bd}$. There are no perfect approximations across this operation, though there are several approximations with bias 2^{-2}.

Immediately we can see the following four-step extension to the existing basic linear approximation:

A	B	C	D	E	
-	-	-	-	e_0	
		↓			1/4
e_0	e_{27}	-	e_0	-	
		↓			1/2
-	e_0	e_{25}	-	e_0	
		↓			1/4
e_0	e_{27}	-	e_{25}	-	
		↓			1/2
-	e_0	e_{25}	-	e_{25}	

The bias for this extension can be computed as 2^{-3}. In extending further we need to approximate across the addition operation in a bit position other than the least significant. We will consider a worst-case scenario for the key values so that the bias of this approximation is perhaps a little less than 2^{-2} for this particular integer addition. We of course expect it to be less.

The following two-step extension allows us to form a ten-step approximation to the steps in Round 1 that holds with a bias of no more than 2^{-5} in the best case (for the analyst) and in the range $(2^{-6}, 2^{-7})$ on average.

A	B	C	D	E
-	e_0	e_{25}	-	e_{25}
		↓		
e_{25}	e_{20}	e_{30}	-	-
		↓		
-	e_{25}	e_{18}	e_{30}	-

Experiments confirm this ten-step linear approximation and for the best key conditions (which hold for one in 2^{25} random keys) the bias was $2^{-5.6}$ and for the average case the bias was $2^{-6.4}$ on average.

We will conservatively use 2^{-5} as the estimate for the bias for this ten-step linear approximation to the steps in Round 3.

Putting things together The ten-step linear approximation we identified for Rounds 2 and 4 is valid over 40 steps of the full SHA-1. Therefore we estimate that in using this approximation the bias as at most $(2^{-6})^4 \times 2^3 = 2^{-21}$. This of course is a highly conservative estimate. Among the many favorable assumptions for the cryptanalyst is that this ten-step linear approximation can be joined to itself. It cannot. Extending this approximation in either direction is likely to provide a severe drop in the exploitable bias of the linear approximation.

For Round 1 we might conservatively estimate that the 20 steps can be approximated using a linear approximation with bias no more than $(2^{-6})^2 \times 2 = 2^{-11}$. Likewise we might estimate that the 20 steps in Round 3 can be approximated using an approximation with bias no more than $(2^{-5})^2 \times 2 = 2^{-9}$.

Under the most favorable conditions for the cryptanalyst (conditions that we believe cannot actually be satisfied) if SHA-1 is to be approximated using a linear approximation then the bias will be no more than $2^{-21} \times 2^{-11} \times 2^{-9} \times 2^2 = 2^{-39}$. Note that the key conditions necessary to give the best bias for the approximations in Rounds 1 and 3 hold exceptionally rarely and so we ignore this case and we deduce that the bias is overwhelmingly likely to fall beneath 2^{-40}. On the other hand, note that the approximation outlined has a zero-bias for many keys and so other approximations would have to be used by the analyst in these cases giving a reduced working bias.

Thus a linear cryptanalytic attack on SHA-1 requiring less than 2^{80} known plaintexts is exceptionally unlikely.

2.2 Differential Cryptanalysis

Differential cryptanalysis is a chosen plaintext attack where the attacker is allowed to choose pairs of plaintexts of his liking, and pairs of a predetermined difference. This difference is often defined as the exclusive-or sum of the two plaintexts. The idea is that the difference in the plaintexts allows the attacker to probabilisticly determine the difference in the intermediate ciphertexts of the cipher. If one is able to determine the difference in the ciphertexts after the last few rounds of the cipher with a high probability, one can often make a search for the key (bits) used in the last round. If these key bits can be determined, the attacker can decrypt all ciphertexts by one round, and repeat the attack on a cipher one round shorter than the original, which is typically easier than the attack on the full cipher.

The main tool in differential cryptanalysis is the *characteristic* and the *differential*. A characteristic is a list of the predicted differences in the ciphertexts after each round of the cipher starting with the plaintext differences, and has a probability connected to it. Characteristics are typically built from concatenating one-round characteristics. The probability of a characteristic is then taken as the product of the probabilities of all involved one-round characteristics. Here one assumes that the involved one-round characteristics are independent, which is most often not exactly the case, but as often it is a good approximation. A differential is a collection of characteristics which have identical starting and ending values. Thus, an s-round differential typically specifies only the difference in the plaintexts and in the ciphertexts after s rounds. The differences in the intermediate ciphertexts are allowed to vary. Thus, the probabilities of a differential are in general higher than for a corresponding characteristic. To enable a successful attack based on differential cryptanalysis, the existence of good characteristics is a necessity, whereas to prove resistance against the attack, one must ensure that all differentials have a low probability. The detection of good differentials has proved to be very difficult for most ciphers, and often one considers only characteristics.

Most often in differential cryptanalysis the definition of difference is defined as the exclusive-or of the two texts involved in a pair. Also for SHA this seems to be the best and obvious definition.

Differentials for SHA What makes differential cryptanalysis difficult on SHA is first, the use of both exclusive-ors and modular additions, a second, the functions f_{if}, f_{xor}, f_{maj}.

First we consider the relation between exclusive-or differences and integer addition. Integer addition of a constant word K to the 32-bit words A and B which only differ in few bits does not necessarily lead to an increase of bit differences in the sums $A+S$ and $B+S$. This may be illustrated by the following special case: Suppose the words A and B only differ in the i-th bit, $i < 31$. Then it holds that with probability $\frac{1}{2}$, $A+S$ and $B+S$ also differ in only the i-th bit. Using formulae (11) one sees that $A+S$ and $B+S$ with probability $\frac{1}{4}$ differ in exactly two (consecutive) bits. There is a special and important case to consider, namely when A and B differ in only the most significant bit, position 31. In that case $A+S$ and $B+S$ differ also only in the most significant bit.

The functions f_{if}, f_{xor}, f_{maj} all operate in the bit-by-bit manner. Thus, one can easily find out how the differences in the outputs of each of the functions behave depending of the differences of the three inputs. Namely, one can consider three inputs of one bit each and an output of one bit. Table 1 shows this for all three functions. The notation of the table is as follows. The first three columns represent the eight possible differences in the one-bit inputs, x, y, z. The next three columns indicate the differences in the outputs of each of the three functions. A '0' denotes that the difference always will be zero, a '1' denotes that the difference always will be one, and a '0/1' denotes that in half the cases the difference will be zero and in the other half of the cases the difference will be one. Note that the function f_{xor} is linear in the inputs, i.e. the difference in the outputs can be determined from the differences in the inputs. However, as we shall see, f_{xor} helps to complicate differential cryptanalysis of SHA.

x	y	z	f_{xor}	f_{if}	f_{maj}
0	0	0	0	0	0
0	0	1	1	0/1	0/1
0	1	0	1	0/1	0/1
0	1	1	0	1	0/1
1	0	0	1	0/1	0/1
1	0	1	0	0/1	0/1
1	1	0	0	0/1	0/1
1	1	1	1	0/1	1

Table 1. Distribution of exor differences through the f-functions.

In the following we consider some characteristics for all rounds and for each of the three different rounds.

All rounds The characteristic of Figure 2 holds with probability one over (any) five steps in any of the four rounds. The question mark (?) indicates an unknown

A	B	C	D	E	*prob*
e_{26}	0	0	0	e_{31}	
		↓			1
0	e_{26}	0	0	0	
		↓			1
?	0	e_{24}	0	0	
		↓			1
?	?	0	e_{24}	0	
		↓			1
?	?	?	0	e_{24}	
		↓			1
?	?	?	?	0	

Table 2. 5-step characteristic.

value. Thus, a pair of texts which differ only in the first words in bit position 26 and in the fifth words in bit position 31, result in texts after five steps which are equal in the fifth words. The difference in the other words of the texts will depend on the particular round considered and of the texts involved.

Rounds 1 and 3 First we consider the five step characteristic of the previous section. With the functions f_{if} and f_{maj} this gives the following characteristic over five steps.

A	B	C	D	E	*prob*
e_{26}	0	0	0	e_{31}	
		↓			1
0	e_{26}	0	0	0	
		↓			$\frac{1}{2}$
0	0	e_{24}	0	0	
		↓			$\frac{1}{2}$
0	0	0	e_{24}	0	
		↓			$\frac{1}{2}$
0	0	0	0	e_{24}	
		↓			$\frac{1}{2}$
e_{24}	0	0	0	0	

This characteristic can be concatenated with a three-step characteristic in the beginning and a two-step characteristic at the end, yielding the following ten-step characteristic.

A	B	C	D	E	*prob*
0	e_1	e_{26}	0	0	
		↓			$\frac{1}{4}$
0	0	e_{31}	e_{26}	0	
		↓			$\frac{1}{4}$
0	0	0	e_{31}	e_{26}	
		↓			$\frac{1}{4}$
e_{26}	0	0	0	e_{31}	
		↓			1
0	e_{26}	0	0	0	
		↓			$\frac{1}{2}$
0	0	e_{24}	0	0	
		↓			$\frac{1}{2}$
0	0	0	e_{24}	0	
		↓			$\frac{1}{2}$
0	0	0	0	e_{24}	
		↓			$\frac{1}{2}$
e_{24}	0	0	0	0	
		↓			$\frac{1}{2}$
e_{29}	e_{24}	0	0	0	
		↓			$\frac{1}{4}$
e_2	e_{29}	e_{22}	0	0	

This ten-step characteristic has a probability of 2^{-13}. As is clearly indicated, extending this characteristic to more steps, e.g., 20, will involve steps with bigger Hamming weights in the differences in the five words than in the first above 10 steps.

We conjecture that the above is one of the characteristics with the highest probability over 10 steps, and that any characteristic over 20 steps of Round 1 or Round 3 will have a probability of less than 2^{-26}.

Rounds 2 and 4 With respect to differential cryptanalysis the function f_{xor} used in Rounds 2 and 4 behaves significantly different from the functions used in Rounds 1 and 3. First note that if we replace all modular additions with exclusive-ors, the steps in Rounds 2 and 4 are linear for exclusive-or differences, in other words, given an input difference one can with probability one determine the output difference after any number of maximum 20 steps. As indicated above, the mixed use of exclusive-ors and modular additions has only little effect for pairs of texts with differences of low Hamming weights. Therefore good characteristics for these steps should have low Hamming weights through as many steps as possible. Consider first the 5-step characteristic of Table 2. The first four steps will evolve as follows.

A	B	C	D	E	*prob*
e_{26}	0	0	0	e_{31}	
		$\downarrow$			1
0	e_{26}	0	0	0	
		$\downarrow$			$\frac{1}{2}$
e_{26}	0	e_{24}	0	0	
		$\downarrow$			$\frac{1}{2}$
$e_{24,31}$	e_{26}	0	e_{24}	0	
		$\downarrow$			$\frac{1}{16}$
$e_{4,24,26,29}$	$e_{24,31}$	e_{24}	0	e_{24}	

Here we have used the notation $e_{a_1,\ldots,a_r}$ for $e_{a_1} \oplus \cdots \oplus e_{a_r}$. It can be seen that for this characteristic the Hamming weights of the differences in the ciphertext words will increase for subsequent steps. Consider as an alternative the following characteristic.

A	B	C	D	E	
e_1	e_3	e_1	e_{11}	$e_{1,3,11}$	
		$\downarrow$			$\frac{1}{16}$
e_6	e_1	e_1	e_1	e_{11}	
		$\downarrow$			$\frac{1}{4}$
e_1	e_6	e_{31}	e_1	e_1	
		$\downarrow$			$\frac{1}{4}$
e_{31}	e_1	e_4	e_{31}	e_1	
		$\downarrow$			$\frac{1}{4}$
e_{31}	e_{31}	e_{31}	e_4	e_{31}	
		$\downarrow$			$\frac{1}{2}$
e_{31}	e_{31}	e_{29}	e_{31}	e_4	
		$\downarrow$			$\frac{1}{4}$
e_{29}	e_{31}	e_{29}	e_{29}	e_{31}	
		$\downarrow$			$\frac{1}{4}$
e_2	e_{29}	e_{29}	e_{29}	e_{29}	
		$\downarrow$			$\frac{1}{4}$
e_7	e_2	e_{27}	e_{29}	e_{29}	
		$\downarrow$			$\frac{1}{16}$
$e_{2,12,27}$	e_7	e_0	e_{27}	e_{29}	
		$\downarrow$			$\frac{1}{32}$
$e_{17,27,29}$	$e_{2,12,27}$	e_5	e_0	e_{27}	

This characteristic was found by a computer search. Of all possible input differences with up to one-bit difference in each of the five input words, totally $33^5 - 1$ characteristics, the last 9 steps of the above characteristic has the lowest Hamming weights in the ciphertexts differences of all steps. For this search we replaced modular additions by exclusive-ors. The nine steps can be concatenated with a one-step characteristic in the beginning, as shown above. In real SHA the probability of these 10 steps is approximately 2^{-26}, where we have used

the above estimates for the behaviour of exclusive-or differences after modular additions. This may **not** give a bound for the best characteristics over 10 steps of SHA, but a complete search seems impossible to implement, moreover it gives sufficient evidence to conclude that there are no high probability characteristics over 20 steps of Rounds 2 and 4. We conjecture that the best such characteristic will have a probability of less than 2^{-32}.

Putting things together Using the estimates for best characteristics for Rounds 1, 2, 3, and 4 of the previous section, we get an estimate of the best characteristic for all 80 steps of SHA, namely $2^{-26} * 2^{-32} * 2^{-26} * 2^{-32} = 2^{-116}$. We stress that this estimate is highly conservative. First of all, the estimates for each round were conservative, and second, there is no guarantee that high probability characteristics for each round in isolation, can be concatenated to the whole cipher. Therefore we conclude that differential cryptanalysis of SHA is likely to require an unrealistic amount of chosen texts if it is possible at all.

3 Conclusions

In the previous section we deduced that a linear cryptanalytic attack on SHA-1 as an encryption function would require at least 2^{80} known plaintexts and that a differential attack would require at least 2^{116} chosen plaintexts. Note that we are explicitly considering constructable linear approximations and differential characteristics. It may well be that there are other approximations and characteristics over SHA-1 that are not revealed by this type of analysis. Instead they would have to be searched for using brute-force. Since there is no known short-cut to such a search this possibility has to be viewed as being so unlikely as to not merit practical consideration.

Our techniques in constructing the approximations and characteristics were *ad hoc*, but based on considerable practical experience. We have been very cautious in our estimates and feel very confident in asserting that a linear or differential cryptanalytic attack using less than 2^{80} plaintext blocks is infeasible. We note that at this point a 160-bit block cipher is beginning to leak plaintext information anyway when used to encrypt this much text with the same key.

Finally we mention that additional cryptanalytic considerations such as linear hulls, multiple linear approximations, and various kinds of differentials are unlikely to make any significant difference to our analysis and estimates. Therefore they make no practical difference to the conclusion we have already drawn.

References

1. F. Chabaud and A. Joux. Differential collisions in SHA-0. In H. Krawczyk, editor, *Advances in Cryptology: CRYPTO'98, LNCS 1462*, pages 56–71. Springer Verlag, 1999.
2. H. Dobbertin. Cryptanalysis of MD5 compress. Presented at the rump session of EUROCRYPT'96, May 1996.

3. H. Dobbertin. Cryptanalysis of MD4. To appear in Journal of Cryptology, 1996.
4. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
5. R.L. Rivest. The MD4 message digest algorithm. In S. Vanstone, editor, *Advances in Cryptology - CRYPTO'90, LNCS 537*, pages 303–311. Springer Verlag, 1991.
6. R.A. Rueppel. *Analysis and Design of Stream Ciphers*. Springer Verlag, 1986.