Finding good differential patterns for attacks on SHA-1

Krystian Matusiewicz and Josef Pieprzyk

Centre for Advanced Computing - Algorithms and Cryptography,
Department of Computing, Macquarie University,
Sydney, NSW 2109, AUSTRALIA
{kmatus,josef}@ics.mq.edu.au

Abstract. In this paper we describe a method of finding differential patterns that may be used to attack reduced versions of SHA-1. We show that the problem of finding optimal differential patterns for SHA-1 is equivalent to the problem of finding minimal weight codeword in a linear code. Finally, we present a number of patterns of different lengths suitable for finding collisions and near-collisions and discuss some bounds on minimal weights of them.

1 Introduction

Most of the modern hash functions used in practice are dedicated hash functions based on the design principles of MD4 [16, 17]. The first cryptanalytic results concerning MD4 appeared only a year after publication of the algorithm [7]. Both MD4 and the improved version of it, called MD5 [18], were first broken by H. Dobbertin [10, 8]. Another hash function based on MD4, called RIPEMD [3] was also shown by Dobbertin [9] to be insecure. Also the shortest variant of HAVAL [23] has been broken by Van Rompay at al. [19]. Recent results obtained by Wang at al. [21, 22] show that is is possible to find collisions for MD4, MD5, HAVAL-128 and RIPEMD within hours on a generic PC. It looks like the message expansion algorithm based on permuting message words and applying them in a different order in each round is a weak point of all these algorithms as it does not provide enough diffusion of differences.

Another group of hash functions are hash functions of the SHA family. Although based on the idea of extended Feistel permutation present in MD, they are equipped with more complex message expansion algorithm. The first function of that family was SHA-0 [11]. It was promptly replaced by an improved version, SHA-1 [15]. Security concerns appeared to be true, as in 1998 Chabaud and Joux presented theoretical attack on SHA-0 [6], which was later implemented and improved allowing for finding an actual collision [13, 12].

Now, one of the most interesting questions in the field of hash functions analysis is how secure is the present standard SHA-1, which is different from SHA-0 by only one rotation in the message expansion process. The same principle used to attack SHA-0 could be applied to construct an attack on SHA-1 provided that there exists good enough differential pattern. Biham and Chen were able to find patterns that allowed for finding collisions for variants reduced to first 34 and 36 steps [2]. The attack can be extended provided that one can find good differential patterns for longer variants of SHA-1.

In this paper we investigate the problem of finding good differential patterns for attacks on SHA-1. First we start with a different presentation of the message expansion algorithm. Next, we show that the problem of finding differential patterns suitable for attacking SHA-1 is a problem of finding low-weight codewords in a linear code.

We present our results of search for the best patterns which can be used in a differential attack and estimate some bounds on the minimal weight of such patterns.

2 Differential attacks on SHA

In this section we briefly recall the structure of SHA-1 and describe the basic framework of differential attacks applicable to SHA-0/1.

2.1 Description of SHA-1 compression function

SHA-1 compression function [15] hashes 512 bit input messages to 160 bit digests. Firstly, 512 bits of the message are divided into 16 words $W_0, W_1, \ldots, W_{15}$ of 32 bits each. The rest of 80 words is generated out of the first 16 words according to the following recurrent formula

$$W_i = ROL^1(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \ \text{ for } \ 16 \leq i \leq 79, \tag{1}$$

where ROL^k means rotation of word by k positions left.

If this is the first application of the compression function , five 32-bit registers A, B, C, D, E are initialized to values $A_0 = 0x67452301$, $B_0 = 0xEFCDAB89$, $C_0 = 0x98BADCFE$, $D_0 = 0x10325476$,
$E_0 = 0xC3D2E1F0$ accordingly.

Next, 80 steps ($i = 0, \ldots, 79$) of the following form are applied:

$$\begin{aligned} A_{i+1} &= ROL^5(A_i) \boxplus f_i(B_i, C_i, D_i) \boxplus E_i \boxplus W_i \boxplus K_i, \\ B_{i+1} &= A_i, \quad C_{i+1} = ROL^{30}(B_i), \\ D_{i+1} &= C_i, \quad E_{i+1} = D_i, \end{aligned} \tag{2}$$

where $\boxplus$ denotes addition modulo 2^{32}, and R_i means the value of the register R after i-th iteration. Functions f_i and constants K_i used in each iteration are defined as

$$f_i(B, C, D) = \begin{cases} BC \vee (\neg B)D & \text{for } \ 0 \leq i \leq 19 \\ B \oplus C \oplus D & \text{for } \ 20 \leq i \leq 39 \\ BC \vee BD \vee CD & \text{for } \ 40 \leq i \leq 59 \\ B \oplus C \oplus D & \text{for } \ 60 \leq i \leq 79 \end{cases}, \tag{3}$$

$$K_i = \begin{cases} 0x5A827999 & \text{for} \quad 0 \leq i \leq 19 \\ 0x6ED9EBA1 & \text{for} \quad 20 \leq i \leq 39 \\ 0x8F1BBCDC & \text{for} \quad 40 \leq i \leq 59 \\ 0xCA62C1D6 & \text{for} \quad 60 \leq i \leq 79 \end{cases} . \tag{4}$$

The output of the compression function is the concatenation of bits of $A_0 \boxplus A_{80}$, $B_0 \boxplus B_{80}$, $C_0 \boxplus C_{80}$, $D_0 \boxplus D_{80}$ and $E_0 \boxplus E_{80}$.

2.2 Differential Attack of Chabaud and Joux

Chabaud and Joux presented in [6] differential attack on SHA-0. The fundamental observation they made is that a change in bit j of word W_i can be corrected by complementary changes in the following bits:

– bit $(j+6) \mod 32$ of word W_{i+1},
– bit j of word W_{i+2},
– bit $(j+30) \mod 32$ of word W_{i+3},
– bit $(j+30) \mod 32$ of word W_{i+4},
– bit $(j+30) \mod 32$ of word W_{i+5},

provided that functions $f_{i+1}, \ldots, f_{i+4}$ and additions $\boxplus$ behave like linear functions, that is a single change in the input to f results in a change of output of f, change in two inputs of f leaves the result unchanged and differences propagate through additions without generating carries. They showed that one bit disturbance can be corrected by such a pattern with probability between 2^{-2} and 2^{-5} depending on functions $f_i, \ldots, f_{i+4}$, if disturbance is introduced in bit $j = 1$.

If disturbance is introduced in position $j \neq 1$, than there is additional factor of 2^{-3} involved, caused by 1/2 chance of inducing carry in additions in steps $i+3$, $i+4$, $i+5$.

The attack is possible due to the property of the message expansion function which does not mix bits in different positions. Thanks to that it was possible to consider message expansion algorithm as a bit-wise. Enumeration of all 2^{16} possible bit patterns in position 1 allowed for choosing disturbance pattern in bit one that led to a global differential pattern δ producing a collision with probability 2^{-61}.

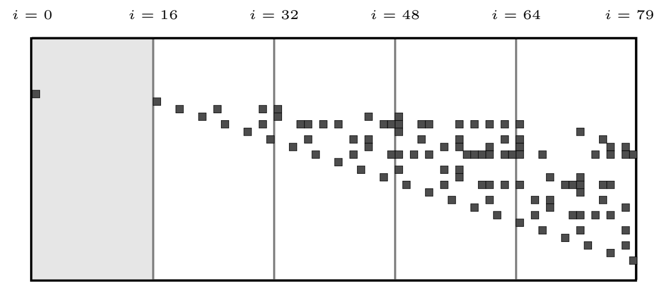
2.3 Improvements

It is possible to improve the attack of Joux and Chabaud by reducing probabilistic behaviour of some initial corrections by a better strategy of selecting messages than picking random ones. Biham and Chen proposed in [1] the method of so called neutral bits. They showed that having a message which behaves correctly for at least 16 first steps after adding a difference δ, it is possible to construct a big set of pairs $(M, M \oplus \delta)$ that have much better probability of successful correction than pairs produced from random messages.

3 Analysis of the message expansion algorithm of SHA-1

Additional rotation in the message expansion formula (1) makes finding corrective patterns used in [6] impossible, because now differences propagate on other positions. For SHA-1, one bit difference in one of the 16 initial blocks propagates to at least 107 bits of the expanded message W. This is illustrated in Fig. 1. However, we were able to find differences that result in only 44 different positions in the expanded message. This suggests that it is interesting to investigate the message expansion algorithm of SHA-1 in a greater detail and check to what extent the differential attack can be applied also to SHA-1.

Fig. 1. Propagation of one bit difference in SHA-1 message expansion.



The important property of the message expansion process given by the formula (1) is that when considered as a function producing 16 new words out of 16 old ones it is a bijection. This implies that it is possible to reconstruct the whole expanded message given any 16 consecutive words of it, in particular the first 16. Moreover, if we consider it on a bit level as a function $A : \mathbb{F}_2^{512} \to \mathbb{F}_2^{512}$, it is easy to see that A is $\mathbb{F}_2$-linear as the only operations used are word rotations (which are permutations of bits) and bitwise XOR operations. Then the expansion of the initial message[1] $m \in \mathbb{F}_2^{512}$ can be expressed as a long vector

$$E_1(m) = \begin{bmatrix} m \\ \hline A(m) \\ \hline A^2(m) \\ \hline A^3(m) \\ \hline A^4(m) \end{bmatrix} \in \mathbb{F}_2^{2560} \ . \tag{5}$$

The set of correction masks is built from a disturbance pattern by rotations and delaying the pattern by 1, 2, . . . , 5 words in the same way as described in [6]. In order to find disturbance patterns which can give rise to correction patterns one has to look for bit patterns $b \in \mathbb{F}_2^{2560}$ that satisfy the following conditions:

[1] we consider m to be a column vector

1. the pattern b has to be of the form (5), i.e. b is the result of the expansion operation,
2. the pattern b ends with $5 \cdot 32 = 160$ zero bits (the last five words are zero), because each disturbance is corrected in the next 5 steps, so no disturbance may occur after the word 74,
3. after delaying a pattern by up to 5 words (that is, shifting bits of b down (right) by $5 \cdot 32 = 160$ positions) the shifted pattern must also be the result of the expansion of its first 512 bits, that is
$$[\underbrace{0 \ldots 0}_{160 \text{ bits}} \, b_0 \, b_1 \, \ldots \, b_{2399}]^T = E_1([0 \ldots 0 \, b_0 \ldots \, b_{351}]^T) \ .$$
4. b has both the minimal Hamming weight and the maximal number of non-zero bits in position 1.

3.1 Basic Construction

Conditions 1– 3 imply that in fact we are looking for longer bit sequences of 85 words such that the first 5 words are zero, the next 11 words are chosen in such a way that while the rest of the words are the result of the expansion of the first 16, the last 5 words are zero again. After denoting the first 5 zero words with indices $-5,\ldots,-1$, in positions $0, \ldots, 79$ we get the disturbance pattern which allows for construction of the corrective pattern.

Using matrix notation, we are looking for a vector $m \in \mathbb{F}_2^{512}$ such that $A^4 m$ has 160 trailing zero bits and also $A^{-1}m$ has 160 trailing zeros. As the transformation A is a bijection, this is equivalent to finding a vector

$$v = [v_0, v_1, \ldots, v_{351}, 0, \ldots, 0]^T \in \mathbb{F}_2^{512} \ ,$$

such that the last 160 bits of $A^{-5}(v)$ contain only zeros, what can be written as

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{351} \\ \hline 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} a_{0,0} & \cdots & \cdots & \cdots & a_{0,511} \\ & & & & \\ \vdots & & & & \vdots \\ & & & & \\ a_{352,0} & \cdots & a_{352,351} & & \\ \vdots & \ddots & \vdots & & \vdots \\ a_{511,0} & \cdots & a_{511,351} & \cdots & a_{511,511} \end{bmatrix} \cdot \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{351} \\ \hline 0 \\ \vdots \\ 0 \end{bmatrix} \ , \qquad (6)$$

where $A^{-5} = (\, a_{i,j} \,)_{\, 0 \le i,j \le 511}$.

This condition means that truncated vectors $\bar{v} = [v_0, v_1, \ldots, v_{351}]^T \in \mathbb{F}_2^{352}$ have to belong to the nullspace of the matrix Ω of the form

$$\Omega = \begin{bmatrix} a_{352,0} & \cdots & a_{352,351} \\ \vdots & \ddots & \vdots \\ a_{511,0} & \cdots & a_{511,351} \end{bmatrix} , \qquad (7)$$

created as a copy of a lower left part of the matrix A^{-5}. It means that the set of all vectors satisfying properties 1– 2 is a linear subspace of $\mathbb{F}_2^{2560}$ with elements of the form

$$c = [\ A^{-4}(v)^T \ || \ A^{-3}(v)^T \ || \ A^{-2}(v)^T \ || \ A^{-1}(v)^T \ || \ v^T\ ]\ , \tag{8}$$

where $v = [\ \bar{v}^T\ ||\ 0 \ldots 0\ ]^T \in \mathbb{F}_2^{512}$ and $\bar{v} \in Ker(\Omega)$.

The set of all such vectors c is in fact a linear code C of length $n = 2560$ and, as we have verified that the rank of the matrix Ω is equal to 192, of dimension $k = 192$.

To maximize the probability of successful correction by the differential pattern, it is necessary to search for the words of minimal Hamming weight and, if possible, for those words with the maximal number of nonzero bits in position 1.

This is essentially the problem of finding minimal distance in a linear code, which is known to be NP-hard [20], so there is no easy way of finding optimal corrective patterns. However, there is a number of probabilistic methods [14, 4] that allow for efficient finding of low-weight codewords in big linear codes in practice.

The second part of the condition 4 can be partially achieved using the fact that the expansion process is invariant in respect of word rotation. The result of the expansion of 16 input words already rotated by a number of bits is the same as the rotation of the result of expansion of 16 words performed without rotation. Thanks to that having a pattern of minimal weight, it is easy to transform it to a pattern with the maximal number of ones in position 1 using word-wise rotation by an appropriate number of positions. Of course, in general this is the problem of finding codewords with minimal weighted weight, however, our experiments show that the simplified approach gives very good results.

3.2 Reduced variants

The generalization of the construction presented above can be applied to find good differential patterns for reduced versions of SHA-1.

Assume that we want to find a differential pattern for SHA-1 reduced to $16 < s \leq 80$ steps (2). Condition 1 implies that the vector $A^{-1}(m)$ has to have 160 trailing zero bits. If we denote the last 160 rows of the matrix A^{-1} as $A^{-1}[352 :: 511]$ then this condition can be written as

$$\mathbb{0} = A^{-1}[352 :: 511] \cdot m\ . \tag{9}$$

To formulate a simple description of constraints inferred from Condition 2, it is convenient to note that the whole message expansion process can be seen as a linear transform $L :\ \mathbb{F}_2^{512} \rightarrow \mathbb{F}_2^{2560}$ represented by a matrix of the form

$$L = \begin{bmatrix} I_{512} \\ \hline A \\ \hline A^2 \\ \hline A^3 \\ \hline A^4 \end{bmatrix}\ ,$$

where I_{512} is an identity matrix and A is a linear transform described before. Now, if we want to find a differential pattern for s steps, 5 words of the expanded message in positions $s-4$, $s-3$, $\ldots$, s have to be zero. In matrix notation, 160 entries in the vector $L \cdot m$ have to be zero, precisely these in positions $(s-4) \cdot 32$, $\ldots$, $s \cdot 32 + 31$. If we denote by $L[32(s-4) :: 32s+31]$ the matrix created by selecting rows of the matrix L with indices $32(s-4), \ldots, 32s+31$, then condition 2. can be written as:

$$\mathbb{0} = L[32(s-4) :: 32s+31] \cdot m \ . \tag{10}$$

Putting together Equations (9) and (10) we obtain the final result. A message $m \in \mathbb{F}_2^{512}$ gives rise to the corrective pattern if and only if $m \in Ker(\Psi_s)$, where

$$\Psi_s = \left[\frac{A^{-1}[352 :: 511]}{L[32(s-4) :: 32s+31]} \right] \tag{11}$$

is a matrix of dimensions 320×512 built by placing rows of $L[32(s-4) :: 32s+31]$ below rows of $A^{-1}[352 :: 511]$.

4 Search for the best patterns

We have shown that the problem of finding disturbance patterns with minimal weights can be seen as a problem of finding minimal weight codewords in a linear code. To find them, we used a simplified version of the algorithm by Leon [14] presented in [5]. We used the parameter $p = 3$ to search for all combinations of up to three rows and for each code we used at least 100 repetitions of the procedure. Obtained results are presented in Table 1. For each variant of SHA-1 (of length 32 - 85) the second column contains the minimal weight of the pattern we could find. Results marked with (*) improve results of Biham and Chen [2]. Patterns we investigated are suitable for attacking only last steps of SHA-1. As the first 20 steps of SHA-1 employ IF boolean function, the first 16 words of disturbance pattern cannot have ones in the same bit position in two consecutive words. Thus for variants longer than 64, where additional constraint is necessary, we gave weights of unrestricted patterns which constitute lower bounds on weights of the patterns compliant to IF condition. We decided to compute lower bound because the algorithm we used ensures that there is no codeword of lower weight with very high probability. This result seems not to extend in a straightforward way to the case of search for restricted patterns and so providing a sound lower bound seems to be more reasonable in this situation.

According to Biham and Chen [2], it is possible to eliminate the probabilistic behaviour of up to 20 first rounds. Thus the third column (denoted wt_{20+}) contains minimal weights of patterns where weights of the first 20 steps are not counted.

We were also interested in patterns which does not allow for finding the full collisions but still are suitable for finding near-collisions as this may possibly lead to an easier way of finding real collisions. To obtain them we relaxed condition that the last five words must contain only zeros and allowed for nonzero entries in one more block. Weights of the best patterns found that way are listed in column wt_n.

Table 1. Hamming weights of the best patterns found. Column wt contains total Hamming weights of patterns, wt_{20+} – weights of patterns with ignored 20 first steps, column wt_n shows total weights of incomplete patterns for near-collisions (patterns ending with only 4 zero blocks).

steps	wt	wt_{20+}	wt_n	steps	wt	wt_{20+}	wt_n	steps	wt	wt_{20+}	wt_n
32	9	2	9	50	35	14	35	68	> 122	> 78	> 90
33	9	2	9	51	35	15	35	69	> 127	> 81	> 127
34	9	2	9	52	35	16	35	70	> 142	> 80	> 124
35	28	4	24	$\boxed{53}$	$\boxed{35}$	16	35	71	> 157	> 94	> 142
36	24	5	24	54	78	36	75	72	> 172	> 93	> 139
37	25	5	25	55	80	39*	73	73	> 139	> 111	> 139
38	30	8	30	56	79	41	72	74	> 139	> 98	> 139
39	39	8*	35	57	72	42	72	75	> 142	> 90	> 142
40	41	11	38	58	73	42	55	76	> 187	> 111	> 187
41	41	12	41	59	91	51	66	77	> 184	> 108	> 184
42	41	13	34	60	66	44	66	78	> 198	> 115	> 177
43	41	17	41	61	66	44	66	79	> 220	> 115	> 220
44	50	15	42	62	66	45	66	80	> 172	> 106	> 172
45	45	15	45	63	107	64	87	81	> 255	> 117	
46	56	23	42	64	> 101	> 60	> 96	82	> 242	> 142	
47	56	24*	35	65	> 113	> 66	> 98	83	> 215	> 163	
48	35	14	35	66	> 98	> 58	> 98	84	> 161	> 101	
49	35	14	35	67	> 127	> 69	> 122	85	> 340	> 177	

It is interesting to see that the minimal weights we were able to find are growing in quite an irregular fashion. In fact, after a rapid jump after reaching 35 steps and a steady growth up till 47 steps, there is an unexpected downfall to weight 35 on step 48. The same pattern, presented in Fig. 3, is suitable for attacks on up to 53 steps. After 53 steps weights get much higher and as we considered patterns without restrictions imposed by the IF function in the first 20 steps of SHA-1, the best pattern for the full SHA-1 will most likely have weight considerably higher than 172.

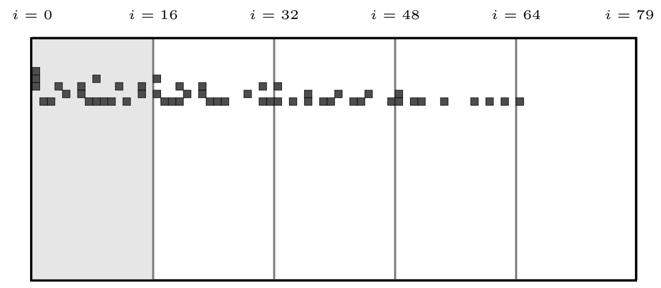
Let us discuss some bounds on the minimal weights of corrective patterns. Consider the inverse of the transformation (1). It can be written as

$$W_i = W_{i+2} \oplus W_{i+8} \oplus W_{i+13} \oplus ROR^1(W_{i+16}), \quad 0 \le i < 64, \qquad (12)$$

where the last 16 words $W_{64},\ldots,W_{79}$ are set arbitrarily.

Although this formula describes essentially the same transform, thanks to the fact that rotation is now applied to only one variable distant by 16 steps, difference propagation of (12) is much worse than the original function. In fact, difference in one bit in one of the last 16 words propagates to only up to 4 different positions and changes only 55 to 82 bits, what is illustrated in Fig. 2. It is interesting to note that this peculiar behaviour does not depend on the rotation amount used in the algorithm but is rather inherent to the structure of recurrence relations similar to (1).

Fig. 2. Inverse propagation of one bit difference applied in the last segment of SHA-1.



To roughly estimate the minimal number of ones in the expansion process we divide the set of ones in two groups: these in the same position that the initial bit and those in different positions. The size of the first group can be easily found experimentally, as there are only 2^{16} of all bit sequences generated by the relation $w_i = w_{i+2} \oplus w_{i+8} \oplus w_{i+13}$, $i > 15$ and much less of them with 5 leading and 5 trailing zeros. Minimal weights of such sequences of different lengths are presented in Table 2. Note that to estimate the number of ones for a differential pattern of length s, minimal weight of a sequence of length $s + 5$ has to be considered with 5 leading and 5 trailing zero bits.

Table 2. Minimal weights of sequences of length $s + 5$ with 5 leading and 5 trailing zeros generated by the formula $w_i = w_{i+2} \oplus w_{i+8} \oplus w_{i+13}$.

s	32–34	35–38	39,40	41	42,43	44–47	48,49	50	51
min. weight	8	9	11	13	11	14	16	17	16

s	52,53	54–56	57–64	65–67	68–71	72	73–75	76,77	78–85
min. weight	17	18	19	23	22	26	24	29	30

The size of the other group of bits cannot be easily estimated. We only can say that it contains at least one element for sequences longer than 16. This makes our estimation work only for not too long variants.

As an example, we can consider the differential pattern for 34 steps. The first set for sequences of length 34 contains at least 8 nonzero bits. The second set must contain at least one bit. Thus, we have shown that the pattern presented in Fig. 3 is the optimal one for that length. This is the same pattern as the one used by Biham and Chen to find collisions for 34 steps of SHA-1 [2].

5 Conclusions

In this paper we presented a way of finding differential patterns for attacks on reduced variants of SHA-1 and we proved that the problem of finding the best pattern is equivalent to the problem of finding minimum weight codeword in a particular linear code. We have found differential patterns for reduced versions of SHA-1 of different lengths between 34 and 85 steps. Our results show that the minimal weights for reduced variants may vary in an unexpected way, but nevertheless the longest variant for which the differential attack of Joux and Chabaud with necessary improvements seems to be possible is the version of the last 53 steps of SHA-1. We presented the actual differential for this variant and improved in a few places weights for shorter variants given by Biham and Chen. We also derived some bounds on minimal weights of differential patterns and proved that the 34 step differential used by Biham and Chen is the optimal one for this length.

References

1. E. Biham and R. Chen. Near collisions of SHA-0. In M. Franklin, editor, *Advances in Cryptology - CRYPTO'04*, volume 3152 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.
2. E. Biham and R. Chen. New results on SHA-0 and SHA-1. Short talk presented at CRYPTO'04 Rump Session, 2004.
3. A. Bosselaers and B. Preneel, editors. *Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation.*, volume 1007 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
4. A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.
5. F. Chabaud. On the security of some cryptosystems based on error-correcting codes. In A. D. Santis, editor, *Advances in Cryptology - EuroCrypt '94*, volume 950 of *Lecture Notes in Computer Science*, pages 131–139, Berlin, 1995. Springer-Verlag.
6. F. Chabaud and A. Joux. Differential collisions in SHA-0. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO'98*, volume 1462 of *LNCS*, pages 56–71. Springer-Verlag, 1998.
7. B. den Boer and A. Bosselaers. An attack of the last two rounds of MD4. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 194–203. Springer-Verlag, 1991.
8. H. Dobbertin. The status of MD5 after a recent attack. *CryptoBytes*, 2(2):1,3–6, 1996.

9. H. Dobbertin. RIPEMD with two-round compress function is not collison free. *Journal of Cryptology*, 10(1):51–70, 1997.
10. H. Dobbertin. Cryptanalysis of MD4. *Journal of Cryptology*, 11(4):253–271, 1998.
11. FIPS 180. Secure hash standard (SHS). National Institute of Standards and Technology, May 1993. Replaced by [15].
12. A. Joux. Collisions in SHA-0. Short talk presented at CRYPTO'04 Rump Session, 2004.
13. C. Lemuet. Collision in SHA-0. `sci.crypt` newsgroup message, Message-ID: `cfg007$1h1b$1@io.uvsq.fr`, 12 August 2004.
14. J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.
15. National Institute of Standards and Technology. Secure hash standard (SHS). FIPS 180-2, August 2002.
16. L. R. Rivest. The MD4 message digest algorithm. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology - CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311. Springer-Verlag, 1991.
17. R. L. Rivest. The MD4 message digest algorithm. Request for Comments (RFC) 1320, Internet Engineering Task Force, April 1992.
18. R. L. Rivest. The MD5 message digest algorithm. Request for Comments (RFC) 1321, Internet Engineering Task Force, April 1992.
19. B. Van Rompay, A. Biryukov, B. Preneel, and J. Vandewalle. Cryptanalysis of 3-pass HAVAL. In C. S. Laih, editor, *Advances in Cryptology - ASIACRYPT'03*, volume 2894 of *Lecture Notes in Computer Science*, pages 228–245. Springer-Verlag, 2003.
20. A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997.
21. X. Wang, X. Lai, D. Feng, and H. Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Short talk presented at CRYPTO'04 Rump Session, 2004.
22. X. Wang, X. Lai, D. Feng, and H. Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199, August 2004. http://eprint.iacr.org/.
23. Y. Zheng and J. a. Pieprzyk. Haval – a one-way hashing algorithm with variable length of output. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology - AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 83–104. Springer-Verlag, 1993.

Fig. 3. Best differential patterns for first 34 steps and last 53 steps of SHA-1.

```
W[ 0]: 00000000000000000000000000000010   W[27]: 00000000000000000000000000000000
W[ 1]: 00000000000000000000000000000000   W[28]: 00000000000000000000000000000000
W[ 2]: 00000000000000000000000000000010   W[29]: 00000000000000000000000000000000
W[ 3]: 00000000000000000000000000000000   W[30]: 01000000000000000000000000000000
W[ 4]: 00000000000000000000000000000010   W[31]: 00000000000000000000000000000000
W[ 5]: 00000000000000000000000000000000
W[ 6]: 00000000000000000000000000000011   W[32]: 00000000000000000000000000000010
W[ 7]: 00000000000000000000000000000000   W[33]: 10000000000000000000000000000000
W[ 8]: 00000000000000000000000000000000   W[34]: 01000000000000000000000000000011
W[ 9]: 00000000000000000000000000000010   W[35]: 00000000000000000000000000000000
W[10]: 00000000000000000000000000000000   W[36]: 00000000000000000000000000000001
W[11]: 00000000000000000000000000000000   W[37]: 10000000000000000000000000000010
W[12]: 00000000000000000000000000000000   W[38]: 10000000000000000000000000000000
W[13]: 00000000000000000000000000000000   W[39]: 00000000000000000000000000000010
W[14]: 00000000000000000000000000000010   W[40]: 00000000000000000000000000000001
W[15]: 00000000000000000000000000000000   W[41]: 00000000000000000000000000000000
                                          W[42]: 10000000000000000000000000000010
W[16]: 00000000000000000000000000000000   W[43]: 00000000000000000000000000000010
W[17]: 00000000000000000000000000000000   W[44]: 10000000000000000000000000000010
W[18]: 00000000000000000000000000000000   W[45]: 00000000000000000000000000000000
W[19]: 00000000000000000000000000000000   W[46]: 10000000000000000000000000000001
W[20]: 00000000000000000000000000000010   W[47]: 00000000000000000000000000000000
W[21]: 00000000000000000000000000000000
W[22]: 00000000000000000000000000000010   W[48]: 10000000000000000000000000000000
W[23]: 00000000000000000000000000000000   W[49]: 00000000000000000000000000000010
W[24]: 00000000000000000000000000000000   W[50]: 10000000000000000000000000000001
W[25]: 00000000000000000000000000000000   W[51]: 00000000000000000000000000000000
W[26]: 00000000000000000000000000000000   W[52]: 00000000000000000000000000000010
W[27]: 00000000000000000000000000000000   W[53]: 00000000000000000000000000000010
W[28]: 00000000000000000000000000000000   W[54]: 00000000000000000000000000000000
W[29]: 00000000000000000000000000000000   W[55]: 00000000000000000000000000000000
W[30]: 00000000000000000000000000000000   W[56]: 00000000000000000000000000000010
W[31]: 00000000000000000000000000000000   W[57]: 00000000000000000000000000000000
                                          W[58]: 00000000000000000000000000000011
W[32]: 00000000000000000000000000000000   W[59]: 00000000000000000000000000000000
W[33]: 00000000000000000000000000000000   W[60]: 00000000000000000000000000000010
                                          W[61]: 00000000000000000000000000000010
                                          W[62]: 00000000000000000000000000000010
                                          W[63]: 00000000000000000000000000000000

                                          W[64]: 00000000000000000000000000000010
                                          W[65]: 00000000000000000000000000000000
                                          W[66]: 00000000000000000000000000000001
                                          W[67]: 00000000000000000000000000000000
                                          W[68]: 00000000000000000000000000000000
                                          W[69]: 00000000000000000000000000000010
                                          W[70]: 00000000000000000000000000000000
                                          W[71]: 00000000000000000000000000000000
                                          W[72]: 00000000000000000000000000000010
                                          W[73]: 00000000000000000000000000000000
                                          W[74]: 00000000000000000000000000000000
                                          W[75]: 00000000000000000000000000000000
                                          W[76]: 00000000000000000000000000000000
                                          W[77]: 00000000000000000000000000000000
                                          W[78]: 00000000000000000000000000000000
                                          W[79]: 00000000000000000000000000000000
```