CHAPTER 9: ANALYSIS OF THE SHA AND SHA-1 HASH ALGORITHMS

9.1 INTRODUCTION

In this chapter the SHA and SHA-1 hash functions are analysed. First the SHA and SHA-1 hash functions are described along with the relevant notation used in this chapter. This is followed by describing the algebraic structure of the message expansion algorithm used by SHA. We then proceed to exploit this algebraic structure of the message expansion algorithm by applying the generalised analysis framework presented in Chapter 8. We show that it is possible to construct collisions for each of the individual rounds of the SHA hash function. The source code that implements the attack is attached in Appendix F. The same techniques are then applied to SHA-1.

9.2 INTRODUCTION TO SHA

SHA is an acronym for Secure Hash Algorithm. SHA and SHA-1 are dedicated hash functions based on the iterative Damgård-Merkle construction [22] [23]. Both of the round functions utilised by these algorithms take a 512 bit input (or a multiple of 512) and produce a 160 bit hash value. SHA was first published as Federal Information Processing Standard 180 (FIPS 180). The secure hash algorithm is based on principles similar to those used in the design of MD4 [10]. SHA-1 is a technical revision of SHA and was published as FIPS 180-1 [13]. It is believed that this revision makes SHA-1 more secure than SHA [13] [50] [59]. SHA and SHA-1 differ from MD4 with regard to the number of rounds used, the size of the hash result and the definition of a single step. A further difference between SHA/SHA-1 and MD4 is the use of a message expansion algorithm instead of re-using the message blocks in different orders in each round. SHA-1 was designed to be both pre-image resistant and collision resistant [13].

9.3 NOTATION

The following notation is used in this chapter.

$\wedge$ = Bitwise AND $\quad$ $\neg$ = Bitwise NOT

$\vee$ = Bitwise OR $\quad$ $\oplus$ = Bitwise XOR.

Let $X^{\lll Z}$ denote the left rotation of X by Z bit positions.

9.4 SHA

Five steps may be identified in the definition of SHA.

1. Message padding.
2. Initialise chaining variables.
3. Perform message expansion.
4. Apply compress function.
5. Update variables.

Step 1 ensures that the message is padded to a multiple of 512 bits. Steps 3 to 5 are repeated for each 512 bit block until the entire padded message has been processed.

9.4.1 Message Padding

The purpose of padding is to produce a $n \cdot 512$ bit message. The message is padded by appending a 1 to the message, followed by m 0's followed by a 64 bit integer. The 64 bit integer is the binary representation of the length of the original message l. If l is less than 2^{32} the first 32 bits of the final 64 bits are zero.

9.4.2 Initialise Chaining Variables

The five chaining variables are initialised to the following hexadecimal values.

$$\begin{aligned} H_0 &= \texttt{0x67452301} \\ H_1 &= \texttt{0xEFCDAB89} \\ H_2 &= \texttt{0x98BADCFE} \\ H_3 &= \texttt{0x10325476} \\ H_4 &= \texttt{0xC3D2E1F0}. \end{aligned}$$

9.4.3 Message Expansion

The message is processed in 512 bit blocks. The padded message M is the concatenation of n blocks of 512 bits. Let $|$ denote concatenation, then:

$$M = M_1|M_2|M_3|\ldots|M_n.$$

Each message block M_i is divided into 16 words W_0, W_1, W_2, ... ,W_{15}. Each word has a length of 32 bits. The execution of the compress function involves 80 steps. The first 16 steps are performed on W_0, W_1, W_2, ... ,W_{15}. The remaining 64 steps are performed on message words W_{16}, W_{17}, W_{18}, ... ,W_{79} which are obtained from the following message expansion algorithm:

$$W_t = W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16} \tag{9.1}$$

with $t = 16, 17, 18, \ldots 79$.

9.4.4 Compress Function

The compress function of SHA is defined by the following procedure:

1. Set $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$ and $E = H_4$.
2. For t = 0 to 79 do

(a) TEMP = $A^{\lll 5} + f_t(B, C, D) + E + W_t + K_t$.

(b) $E = D, D = C, C = B^{\lll 30}, B = A, A = TEMP$

with the constant K_t defined in hexadecimal as:

$$\begin{aligned} K_t &= \texttt{0x5A827999} \quad (0 \leq t \leq 19) \\ K_t &= \texttt{0x6ED9EBA1} \quad (20 \leq t \leq 39) \\ K_t &= \texttt{0x8F1BBCDC} \quad (40 \leq t \leq 59) \\ K_t &= \texttt{0xCA62C1D6} \quad (60 \leq t \leq 79) \end{aligned}$$

and the round function f_t defined as:

$$\begin{aligned} f_t &= (B \wedge C) \vee (\neg B \wedge D) && (0 \leq t \leq 19) \\ f_t &= B \oplus C \oplus D && (20 \leq t \leq 39) \\ f_t &= (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) && (40 \leq t \leq 59) \\ f_t &= B \oplus C \oplus D && (60 \leq t \leq 79) \end{aligned} .$$

Figure 9.1 shows a graphical representation of a single step of the SHA and SHA-1 round function.

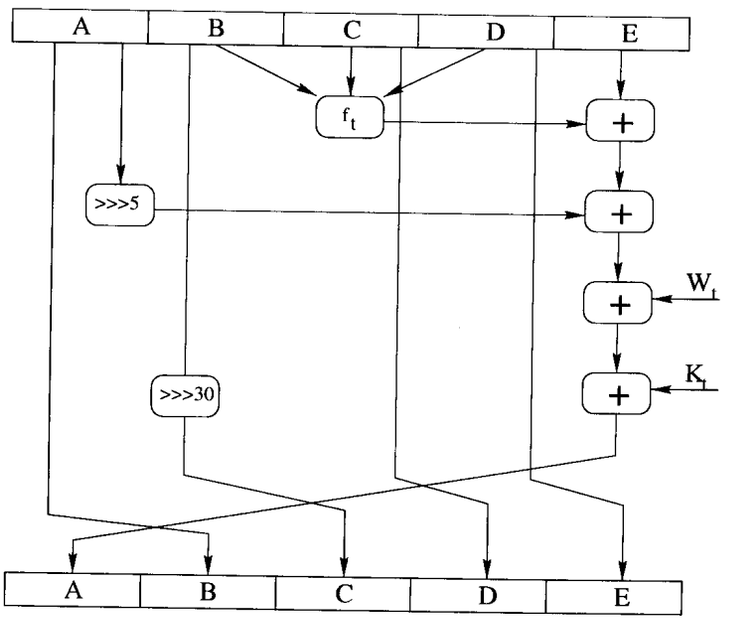

Figure 9.1: Single Step in Round Function: SHA and SHA-1

9.4.5 Update Chaining Variables

Upon completion of the compress function the chaining variables are updated as shown below:

$$\begin{aligned} H_0 &= H_0 + A \\ H_1 &= H_1 + B \\ H_2 &= H_2 + C \\ H_3 &= H_3 + D \\ H_4 &= H_4 + E \end{aligned}$$

Once the n'th 512 bit block has been processed the resulting values of the chaining variables serve as the hash result of the message.

9.5 SHA-1

SHA-1 is identical to SHA in all respects except for the message expansion algorithm.

9.5.1 Message Expansion

The message expansion algorithm used in SHA-1 is defined as the message expansion algorithm defined by expression 9.1 with the addition of a left rotation by one bit position. The message expansion algorithm as used in SHA-1 is defined by expression 9.2.

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})^{\lll 1}. \tag{9.2}$$

This modification "corrects a technical flaw that made the standard less secure than had been thought" [50].

9.6 ANALYSIS OF SHA AND SHA-1

In this chapter the basic elements encountered in the attacks on MD4 and MD5 as formulated by Dobbertin are applied to SHA and SHA-1. A direct application of the attacks formulated

by Dobbertin requires the establishment of a set of difference equations, followed by a solution of these equations (see Chapter 8). The use of message expansion algorithms in SHA and SHA-1 instead of the permutation of message words in consecutive rounds prevents an attacker from deriving sets of difference equations according to the principles laid down in Chapter 8. Thus when dealing with message expansion algorithms the approach described in Chapter 8 should be modified. This chapter describes these modifications. Specific attention is given to the properties of the message expansion algorithms. It is shown that the modified approach may be used to construct a collision for the first round (first twenty steps) of SHA. Based on this attack, an attack on the first two rounds of SHA is proposed. Certain elements (but not all) of the proposed attack are confirmed. The extent to which these attacks are applicable to SHA-1 is considered.

9.7 SHA

The secure hash algorithm (SHA) is described in Section 9.4. In this section certain properties of the message expansion algorithm is considered. It is shown that these properties may be exploited to construct sets of difference equations which are readily solved. A solution to these difference equations results in the construction of collisions for the first round of SHA. An attack which exploits the properties of the message expansion algorithm is then proposed for the first two rounds of SHA.

9.7.1 Message Expansion Algorithm

The message expansion algorithm used in SHA is presented in Chapter 9. The algorithm expands a 16 word input to 64 words. The expanded message is concatenated to the original message to form an 80 word message block. Remember that the message expansion algorithm is given by:

$$W_t = W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}$$

where all message words, W_t, are 32 bit variables. It is observed that this expansion algorithm is in effect 32 identical linear feedback shift registers which operate in parallel. This observation may be represented graphically as shown in Figure 9.2.

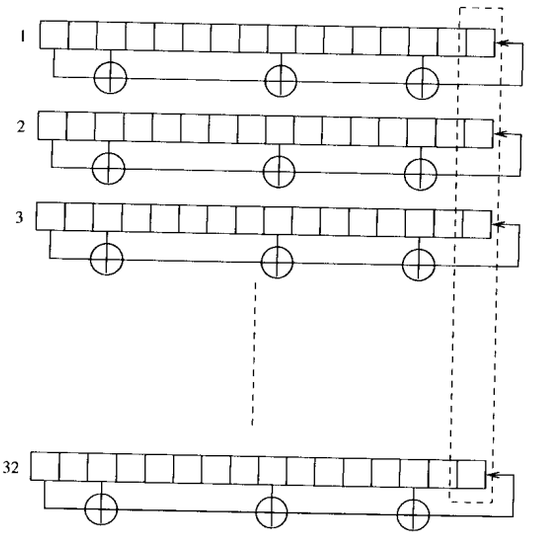

Figure 9.2: Message Expansion Algorithm: SHA

The connection polynomial, $f(x)$, is given by:

$$f(x) = x^{16} + x^{13} + x^8 + x^2 + 1. \tag{9.3}$$

It may be shown that $f(x)$ is primitive. It is known that there exists

$$\frac{\phi(p^m - 1)}{m}$$

monic primitive polynomials of degree m in $Z_p[x]$ ($\phi(n)$ is the Euler function) [59]. For the case in hand this corresponds to 2048 possible monic primitive polynomials in $Z_2[x]$. The reason for the specific choice of $f(x)$ has not been disclosed, but at least two arguments in favour of using $f(x)$ have been found. The first argument deals with performance. The polynomial $f(x)$ has a low weight (5), and consequently requires fewer instructions to expand the message. The second argument deals with the positioning of the non-zero coefficients in the polynomial. The non-zero coefficients should be spread as evenly as possible over the polynomial. This requirement ensures that each word is used in the expansion within a small number of steps. This makes it more difficult for an attacker to hide or minimise the effect of a specific word. There are 52 primitive polynomials of degree 16 with a weight of 5. It is found that the maximum number of consecutive coefficients which are equal to zero varies between 5 and 10. There are only 16 primitive polynomials with a maximum number

of consecutive zero coefficients equal to zero. They are:

$$f_1(x) = x^{16} + x^{10} + x^5 + x^3 + 1 \quad (9.4)$$
$$f_2(x) = x^{16} + x^{10} + x^7 + x + 1 \quad (9.5)$$
$$f_3(x) = x^{16} + x^{10} + x^7 + x^3 + 1 \quad (9.6)$$
$$f_4(x) = x^{16} + x^{10} + x^7 + x^4 + 1 \quad (9.7)$$
$$f_5(x) = x^{16} + x^{10} + x^7 + x^6 + 1 \quad (9.8)$$
$$f_6(x) = x^{16} + x^{10} + x^9 + x^6 + 1 \quad (9.9)$$
$$f_7(x) = x^{16} + x^{12} + x^6 + x + 1 \quad (9.10)$$
$$f_8(x) = x^{16} + x^{12} + x^7 + x + 1 \quad (9.11)$$
$$f_9(x) = x^{16} + x^{12} + x^9 + x^6 + 1 \quad (9.12)$$
$$f_{10}(x) = x^{16} + x^{13} + x^8 + x^2 + 1 \quad (9.13)$$
$$f_{11}(x) = x^{16} + x^{13} + x^9 + x^6 + 1 \quad (9.14)$$
$$f_{12}(x) = x^{16} + x^{13} + x^{11} + x^6 + 1 \quad (9.15)$$
$$f_{13}(x) = x^{16} + x^{14} + x^8 + x^3 + 1 \quad (9.16)$$
$$f_{14}(x) = x^{16} + x^{15} + x^9 + x^4 + 1 \quad (9.17)$$
$$f_{15}(x) = x^{16} + x^{15} + x^9 + x^6 + 1 \quad (9.18)$$
$$f_{16}(x) = x^{16} + x^{15} + x^{10} + x^4 + 1 \quad (9.19)$$
$$(9.20)$$

Eight of these polynomials has two consecutive non-zero coefficients ($f_2(x)$, $f_5(x)$, $f_6(x)$, $f_7(x)$, $f_8(x)$, $f_{14}(x)$, $f_{15}(x)$ and $f_{16}(x)$). The primitive polynomial used in the message expansion algorithm of SHA (and SHA-1) is found among the remaining eight polynomials ($f_{10}(x)$).

It is observed that the message block of 512 bits is divided into 16 32-bit words and that each round requires 20 steps. This differs from hash functions such as MD4 and MD5 where the number of message words in a message block are equal to the number of steps in each round. Before presenting a motivation for this design choice consider Proposition 1.

Proposition 1 *If the message expansion algorithm defined for SHA is used, it is possible to construct two distinct messages which, after expansion, are identical in 15 consecutive expanded words.*

Proof. Consider a single linear feedback shift register with a connection polynomial $f(x)$ of degree n which is monic and primitive in $Z_2[x]$. It is known that all non-zero sequences generated by such a linear feedback shift register have $n-1$ consecutive zeros. In addition it is known that any possible output sequence of such a linear feedback shift register is a cyclic shift of every other possible output sequence of the same linear feedback shift register [60]. Thus there exists a non-zero sequence, $\boldsymbol{a}$, generated by the linear feedback shift register with feedback polynomial $f(x)$ such that $\boldsymbol{a}$ has $n-1$ consecutive zeros starting at a specified position in the sequence $\boldsymbol{a}$.

Let $f(x) = c_0 + c_1x + c_2x^2 \ldots + c_nx^n$ with coefficients from $GF(2)$. Then the companion matrix of the polynomial $f(x)$ is the $n \times n$ matrix, $\boldsymbol{C}$, which has 1's in the diagonal above the main diagonal and n'th row entries c_0, c_1, c_2 $\ldots$ c_{n-1} [61]. Then the t'th state of the linear feedback shift register is given by:

$$a(t) = \boldsymbol{C}^t a(0)$$

with $a(t)$ a $1 \times n$ vector representing the state of the linear feedback shift register. The elements of the vector $a(0)$ represents the initial state of the shift register. Let the sequence $\boldsymbol{a}$ be the concatenation (denoted by $|$) of the n'th element of the vector $a(t)$ for all values of $t > 0$. Thus:

$$\boldsymbol{a} = a_n(1)|a_n(2)|a_n(3)|\ldots|a_n(t).$$

Consider two sequences $\boldsymbol{a^1}$ and $\boldsymbol{a^2}$ defined as:

$$\begin{aligned} \boldsymbol{a^1} &= a_n^1(1)|a_n^1(2)|a_n^1(3)|\ldots|a_n^1(t) \\ \boldsymbol{a^2} &= a_n^2(1)|a_n^2(2)|a_n^2(3)|\ldots|a_n^2(t). \end{aligned}$$

The states, $a^1(t)$ and $a^2(t)$, of the linear feedback shift registers which generates the sequences $\boldsymbol{a^1}$ and $\boldsymbol{a^2}$ are given by:

$$\begin{aligned} a^1(t) &= \boldsymbol{C}^t a^1(0) \\ a^2(t) &= \boldsymbol{C}^t a^2(0). \end{aligned}$$

The summation over $GF(2)$ of the states of the linear feedback shift registers (and implicitly

the summation of the sequences $\boldsymbol{a^1}$ and $\boldsymbol{a^2}$) is given by:

$$\begin{aligned} a^1(t) \oplus a^2(t) &= \boldsymbol{C}^t a^1(0) \oplus \boldsymbol{C}^t a^2(0) \\ &= \boldsymbol{C}^t (a^1(0) \oplus a^2(0)). \end{aligned}$$

If

$$a^1(0) \oplus a^2(0) = a(0) \tag{9.21}$$

then

$$\begin{aligned} a^1(t) \oplus a^2(t) &= \boldsymbol{C}^t (a(0)) \\ &= a(t) \end{aligned}$$

which implies that:

$$\boldsymbol{a} = \boldsymbol{a^1} \oplus \boldsymbol{a^2}.$$

The message expansion algorithm used by SHA consists of 32 identical linear feedback shift registers applied in parallel. Let the initial conditions $a^1(0)$ and $a^2(0)$ for all 32 linear feedback shift registers represent two distinct messages. All elements in $\boldsymbol{a^1}$ and $\boldsymbol{a^2}$ are taken as the result of the message expansion algorithm. Find values for $a^1(0)$ and $a^2(0)$ such that expression (9.21) is satisfied for all 32 registers. Then, upon expansion, the message words will be identical in $n - 1$ consecutive bit positions. The primitive polynomial used by the linear feedback shift register in the message expansion algorithm in SHA has degree 16. It is therefore possible to construct two messages which upon expansion will be identical in 15 consecutive words. Thus the proposition is shown to be true. ■

With Proposition 1 in hand the use of 20 steps in a round rather than 16 becomes obvious. If the number of steps in each round is set to 16 instead of 20 and the number of rounds are retained it would be possible to construct two distinct messages which, in one of the four rounds, would differ in only one position, thus effectively stripping a round from the hash function. By defining 20 steps in each round this attack is prevented. However if the number of steps are limited to 16, and the number of rounds is extended from four to five, the total number of steps required by the hash function remain at 80.

Thus it appears that the message expansion algorithm was chosen with specific aims, and that the number of steps in each round was chosen to complement the message expansion

algorithm. One property, and a potential weakness, of the message expansion algorithm is the fact that if two messages differ only in a single message word, and if that difference is limited to a single bit position, the words obtained from the message expansion algorithm will, at most, differ in the same bit position. Even if the differences occur in more than one message word, as long as these differences occur in the same bit position, the expanded words would at most differ in the same bit positions. This property is considered a possible salient point and the extent to which this may be exploited is considered in the next section.

9.7.2 Difference Equations

In this section specific attention is given to the exploitation of the message expansion algorithm in order to obtain sets of differential equations.

It is possible to obtain a set of difference equations over all 80 steps of the compress function of SHA. As remarked in Chapter 8 this is impractical due to the large number of interrelated equations which have to be solved. An attempt should therefore be made to reduce the number of difference equations. It is observed that by direct application of Proposition 1 the number of difference equations may be reduced from 80 to 65. This reduction is obtainable by requiring that the last 15 words resulting from the message expansion algorithm should be identical. Although this reduces the effort required to find a collision, the effort required to solve this set of equations is unknown.

Before proceeding with the analysis of SHA and the construction of difference equations it is convenient to state the following definition.

Definition 3 *A difference pattern is the sequence generated by $M_i - \tilde{M}_i$ for all i. M_i and $\tilde{M}_i$ represents the first message and second message words at a specific step i of the dedicated hash function.*

The difference operator may be either the difference $\bmod 2$ or the difference $\bmod 2^{32}$. In order to illustrate the derivation of a set of difference equations consider only the first round. It is readily observed that for six consecutive steps, the following difference pattern (mod

2^{32}) allows the derivation of a set of difference equations which are easily solved:

$$\begin{aligned} M_i - \tilde{M}_i &= 1 \\ M_{i+1} - \tilde{M}_{i+1} &= -1 \\ M_{i+2} - \tilde{M}_{i+2} &= -1 \\ M_{i+3} - \tilde{M}_{i+3} &= 0 \\ M_{i+4} - \tilde{M}_{i+4} &= 0 \\ M_{i+5} - \tilde{M}_{i+5} &= -1 \end{aligned}$$

The difference $\bmod 2^{32}$ is chosen to be either 1,0 or -1 since these differences may be written in terms of the rotation invariant integers 0 and -1. By limiting all differences to the LSB a difference pattern $\bmod 2$ may be obtained. The difference pattern $\bmod 2$ is given by:

$$\begin{aligned} M_i \oplus \tilde{M}_i &= 1 \\ M_{i+1} \oplus \tilde{M}_{i+1} &= 1 \\ M_{i+2} \oplus \tilde{M}_{i+2} &= 1 \\ M_{i+3} \oplus \tilde{M}_{i+3} &= 0 \\ M_{i+4} \oplus \tilde{M}_{i+4} &= 0 \\ M_{i+5} \oplus \tilde{M}_{i+5} &= 1 \end{aligned}$$

The set of difference equations corresponding to this difference pattern is given by:

$$A_{i+1} - \tilde{A}_{i+1} = M_i - \tilde{M}_i \tag{9.22}$$

$$0 = A_{i+1}^{\lll 5} - \tilde{A}_{i+1}^{\lll 5} + M_{i+1} - \tilde{M}_{i+1} \tag{9.23}$$

$$0 = M_{i+2} - \tilde{M}_{i+2} + f(B_{i+2}, C_{i+2}, D_{i+2}) - f(\tilde{B}_{i+2}, C_{i+2}, D_{i+2}) \tag{9.24}$$

$$0 = f(B_{i+3}, C_{i+3}, D_{i+3}) - f(B_{i+3}, \tilde{C}_{i+3}, D_{i+3}) \tag{9.25}$$

$$0 = f(B_{i+4}, C_{i+4}, D_{i+4}) - f(B_{i+4}, C_{i+4}, \tilde{D}_{i+4}) \tag{9.26}$$

$$0 = M_{i+5} - \tilde{M}_{i+5} + E_{i+5} - \tilde{E}_{i+5} \tag{9.27}$$

The Boolean mapping $f()$ is the mapping defined for the first round of SHA. The variables are updated as described in Section 9.4. This set of expressions may be solved by setting the chaining variables C_2, D_2, B_3 and B_4 to appropriate values.

There exists many difference patterns for the first round which yields sets of solvable difference equations. It now remains to determine if the difference pattern leading to this set of equations can be found in the first round. It is now appropriate to state Corollary 2.

Corollary 2 *Each sequence,* $\boldsymbol{a}$*, generated by a linear feedback shift register with feedback polynomial* $f(x)$ *represents a difference pattern which may be generated by two distinct sequences,* $\boldsymbol{a^1}$ *and* $\boldsymbol{a^2}$*.*

Proof. It is shown in the proof of Proposition 1 that:

$$\boldsymbol{a^1} \oplus \boldsymbol{a^2} = \boldsymbol{a}$$

if

$$a^1(0) \oplus a^2(0) = a(0). \tag{9.28}$$

This implies that if the k'th element of the sequence $a_k = 0$, $a_k^1 = a_k^2$. Conversely it implies that if $a_k = 1$, $a_k^1 \neq a_k^2$. Thus the sequence $\boldsymbol{a}$ represents the difference between the two sequences $\boldsymbol{a^1}$ and $\boldsymbol{a^2}$ if condition (9.28) holds. ■

As before, all differences are limited to the LSB of the message words. This causes all differences in the expanded message to be generated by a single linear feedback shift register. An attacker may now search for the desired difference pattern $\bmod 2$. Once the difference pattern $\bmod 2$ is found it is expanded to the difference pattern $\bmod 2^{32}$ through the application of Corollary 2. The attacker is free to choose values for $a^1(0)$ and $a^2(0)$ as long as expression (9.28) holds. A difference pattern $\bmod 2^{32}$ may be established by remarking that for $a_k^1 \oplus a_k^2 = 1$, either $a_k^1 = 1$ and $a_k^2 = 0$ or $a_k^1 = 0$ and $a_k^2 = 1$. This implies that if $a_k^1 \oplus a_k^2 = 1$, either $a_k^1 - a_k^2 = -1$ or $a_k^1 - a_k^2 = 1$. According to Corollary 2 the attacker may choose whether $a_k^1 - a_k^1 = 1$ or $a_k^1 - a_k^1 = -1$ for $k < n$.

Thus, the attacker searches for an initial value which, in combination with the expanded message word, would allow the establishment of the set of difference equations defined by equations (9.22) to (9.27) as the only set of difference equations for the first round. If such a pattern is found, it is expanded to the desired difference pattern $\bmod 2^{32}$ by applying Corollary 2. It is found that there exists only one difference pattern which allows equations (9.22) to (9.27) to be the only set of difference equations in the first round. The difference

pattern mod2 is given by:

$$M_{11} \oplus \tilde{M}_{11} = 1 \tag{9.29}$$
$$M_{12} \oplus \tilde{M}_{12} = 1 \tag{9.30}$$
$$M_{13} \oplus \tilde{M}_{13} = 1 \tag{9.31}$$
$$M_{14} \oplus \tilde{M}_{14} = 0 \tag{9.32}$$
$$M_{15} \oplus \tilde{M}_{15} = 0 \tag{9.33}$$
$$M_{16} \oplus \tilde{M}_{16} = 1. \tag{9.34}$$

It is observed that M_{11} to M_{15} forms part of the initial value and according to Corollary 2 any combination of the LSB's of M_{11} to M_{15} may be chosen in an attempt to obtain the desired difference pattern mod 2^{32} for message words M_{11} to M_{16}. There exists a combination which allows the attacker to find the required difference mod 2^{32}.

The difference pattern defined by equation (9.29) to (9.33) results in the set of difference equations shown below:

$$A_{12} - \tilde{A}_{12} = M_{11} - \tilde{M}_{11} \tag{9.35}$$
$$0 = A_{12}^{\lll 5} - \tilde{A}_{12}^{\lll 5} + M_{12} - \tilde{M}_{12} \tag{9.36}$$
$$0 = M_{13} - \tilde{M}_{13} + f(B_{13}, C_{13}, D_{13}) - f(\tilde{B}_{13}, C_{13}, D_{13}) \tag{9.37}$$
$$0 = f(B_{14}, C_{14}, D_{14}) - f(B_{14}, \tilde{C}_{14}, D_{14}) \tag{9.38}$$
$$0 = f(B_{15}, C_{15}, D_{15}) - f(B_{15}, C_{15}, \tilde{D}_{15}) \tag{9.39}$$
$$0 = M_{16} - \tilde{M}_{16} + E_{16} - \tilde{E}_{16}. \tag{9.40}$$

Updating of chaining variables are performed as specified in Section 9.4. This set of equations implies that:

$$A_{12} - \tilde{A}_{12} = 1.$$

Due to the effect of rotation, the values for A_{12} and $\tilde{A}_{12}$ are chosen to be rotation invariant.

Thus in hexadecimal notation:

$$A_{12} = \texttt{0x00000000} \tag{9.41}$$

$$\tilde{A}_{12} = \texttt{0xFFFFFFFF} \tag{9.42}$$

$$\tag{9.43}$$

These choices for A_{12} and $\tilde{A}_{12}$ ensures that equation (9.36) is satisfied. Equation (9.37) is satisfied if:

$$f(B_{13}, C_{13}, D_{13}) - f(\tilde{B}_{13}, C_{13}, D_{13}) = -1.$$

Due to the choices for A_{12} and $\tilde{A}_{12}$ this condition is satisfied if:

$$D_{13} - C_{13} = -1$$

In order to solve equation (9.38),

$$f(B_{14}, C_{14}, D_{14}) - f(B_{14}, \tilde{C}_{14}, D_{14}) = 0.$$

By setting $B_{14} = 0$, equation (9.38) is satisfied. Likewise equation (9.39) is satisfied if $B_{15} = -1$. Equation (9.40) is automatically satisfied due to the choices initially made for A_{12} and $\tilde{A}_{12}$.

It is noted that this is not the only technique which results in a solution for the set of difference equations given by (9.35) to (9.40). Other more elaborate techniques exist, but are slower and do not guarantee that a solution is obtained.

Given appropriate choices for the chaining variables in question, it is possible to reconstruct the message. For the initial values specified for SHA, the following messages result in a collision for the first round of the compress function of SHA.

$$M_0 = \texttt{0x20760CF1} \qquad M_8 = \texttt{0xF4AD6572}$$
$$M_1 = \texttt{0x0C1F1475} \qquad M_9 = \texttt{0x5F059EA3}$$
$$M_2 = \texttt{0x56139C91} \qquad M_{10} = \texttt{0x050A6650}$$
$$M_3 = \texttt{0xA904D458} \qquad M_{11} = \texttt{0x5279A115}$$
$$M_4 = \texttt{0x07F3FF32} \qquad M_{12} = \texttt{0xBB4E5B88}$$
$$M_5 = \texttt{0x69B971AD} \qquad M_{13} = \texttt{0x724D80BA}$$
$$M_6 = \texttt{0x13E8DD88} \qquad M_{14} = \texttt{0x438ECCB0}$$
$$M_7 = \texttt{0x40CA61AC} \qquad M_{15} = \texttt{0xA1EDDF3D}$$

The alternative message is identical to the above message except for:

$$\tilde{M}_{11} = \texttt{0x5279A114}$$
$$\tilde{M}_{12} = \texttt{0xBB4E5B89}$$
$$\tilde{M}_{13} = \texttt{0x724D80BB}.$$

The common message digest for the first round of SHA is given by:

Common Digest = `0x5C2E4C26 0x494479D5 0x828CE366 0xC45C9D77 0xE437389D`.

An implementation of this attack on the first round of SHA is attached as Appendix F. Attacks similar to that described above may be readily applied to rounds two, three and four if these rounds are considered individually.

9.7.3 Extended Attack

The attack presented above may be extended to other difference patterns. This allows the formulation of the following attack on the first round of SHA.

First a number of relatively short difference patterns which result in inner collisions are obtained. It should then be determined if any of these short difference patterns occur in the first round of SHA. If none of these patterns are found in the first round, a search should

be conducted for concatenations of these difference patterns. If a concatenation of these difference patterns are found, it is known that the resulting set of difference equations may be solved and a high probability exist that a message may be reconstructed which would result in a collision for the first round. Note that the construction of a message which results in a collision is not assured. This is due to the fact that even though the difference pattern results in a set of solvable difference equations, the specific choices made to solve these equations may contradict the bounds incurred by the message expansion algorithm.

9.7.4 Proposed Attack

It may be possible to extend the attack described in the previous section to the second and possibly third and fourth rounds. As before, short difference patterns which results in inner collisions should be obtained for all the rounds in question. The output of the message expansion algorithm should then be searched for the concatenation of a number of these patterns. If a difference pattern is found which is composed from the concatenation of the shorter difference pattern, it may be possible to find a solution to the set of difference equations. If a solution is obtained, it should be verified if a message may be reconstructed which would result in a collision. As before the specific choices made to solve these equations may contradict the bounds incurred by the message expansion algorithm.

An application of this proposed attack showed that it is possible to find difference patterns which are solvable. Unfortunately it was found that the choices made for a number of the chaining variables which allow solutions to be found for the short difference equations leaves a limited number of degrees of freedom. This limits the attackers ability to reconstruct a message which results in a collision for more than one round. If more sophisticated techniques are found to solve the sets of difference equations, fewer explicit choices would have to be made and it may become possible to find solutions to the sets of difference equations which allows the construction of messages which result in collisions for two or more rounds.

9.8 SHA-1

As remarked in Chapter 9, the only difference between SHA and SHA-1 lies in the message expansion algorithm. In this section the message expansion algorithm used in SHA-1 is considered and a number of its characteristics are discussed.

9.8.1 Message Expansion Algorithm

The message expansion algorithm used in SHA-1 is defined by:

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})^{\lll 1}. \tag{9.44}$$

It is noted that the only difference between the message expansion algorithms used in SHA and SHA-1 is the addition of a rotation by one bit position. A graphic representation of the message expansion algorithm used in SHA-1 is shown in Figure 9.3

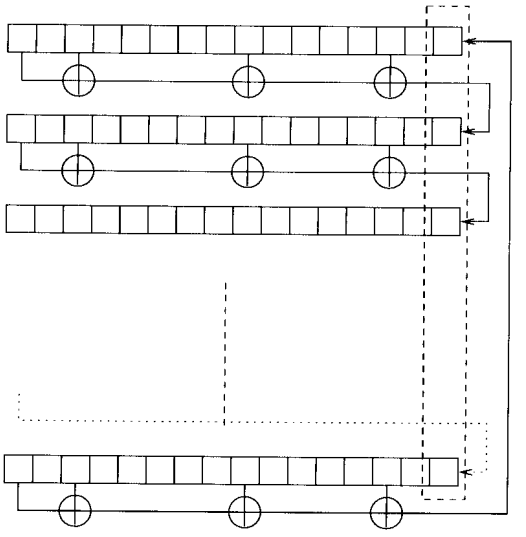

Figure 9.3: Message Expansion Algorithm: SHA-1

The rotation operator introduces diffusion in the message expansion algorithm. A difference introduced in a single bit position is no longer limited to the same bit position in the expanded message (as is the case for SHA), but is spread over a number of bit positions. As the original message is expanded a larger number of bit positions are affected by changing a single bit in the message word. Thus the addition of diffusion to the message expansion algorithm is believed to increase the security offered by SHA-1.

The use of the rotation operator makes it difficult to specify a difference pattern which is easily related to an initial message. Thus the analysis of SHA presented in Section 9.7 is not directly applicable to SHA-1. It is believed that the addition of the rotation operator to the message expansion algorithm makes it considerably more difficult to obtain and exploit difference equations. At present no analysis of the message expansion algorithm used by SHA-1 has been published in the open literature.

9.9 CONCLUSION

The analysis presented in this chapter leads to the conclusion that SHA-1 is more secure than the original SHA. The additional security of SHA-1 is derived solely from the modified message expansion algorithm. It was shown that it is possible to exploit the characteristics of the message algorithm defined for SHA by constructing a collision for the first round of SHA. In addition an attack on more than one round of SHA is proposed. It may be argued that the ease with which the message expansion algorithm used in SHA is manipulated may have served as one of the reasons for the modification to the message expansion algorithm used in SHA-1.

Additional factors which complicates the analysis of SHA and SHA-1 have been found. Specifically the method used for updating the chaining variables and the use of a rotation over 30 bits for chaining variables C_i, D_i and E_i contributes to the difficulty of solving sets of difference equations. The method used for updating the chaining variables ensures that a difference introduced in a message propagates to each of the chaining variables. This is not the case for MD4 and MD5 where a difference may be manipulated to appear only in certain selected chaining variables. An additional difficulty is the use of rotation over chaining variables C_i, D_i and E_i. A single rotation is introduced when C_i is updated. This rotated value is re-used in chaining variables D_i and E_i. If a set of difference equations is obtained, the use of the rotation limits the number of choices which could be made for the chaining variables involved. This in turn reduces the number of solutions which are easily obtained for the difference equations. It is these factors which, at present, prevent the proposed attack on the first two rounds of SHA to be successful. If improved solution techniques becomes available, it may become possible to execute the proposed attack and be able to construct messages which result in collisions for the first two rounds of SHA.

CHAPTER 10: ANALYSIS OF THE HAVAL HASH ALGORITHM

10.1 INTRODUCTION

In this chapter the HAVAL hash function is analysed within the generalised framework presented in Chapter 8. First we describe the HAVAL hash function and the relevant notation needed in this chapter. We then show how the generalised attack formulated in Chapter 8 can be applied to the last two rounds of three round HAVAL to establish a collision. This is the first published cryptanalytical result for the HAVAL hash function. The source code that implements the attack is attached in Appendix G.

10.2 INTRODUCTION TO HAVAL

HAVAL is an iterated hash function based on the Dåmgard-Merkle scheme. The HAVAL construction is closely related to the MD4 family of hash functions. HAVAL was designed by Zheng, Pieprzyk and Seberry in 1994 [62]. The HAVAL hash function specification includes 15 variations [62]. These variations are based on the number of iterations (or rounds) used by the round function (3, 4 or 5) as well as the number bits used as output (128, 160, 192, 224 or 256). The round function of HAVAL takes message blocks in multiples of 1024 bits and produces an output of 256 bits which can then be reduced to 128, 160, 192, 224 or 256 bits, depending on the security requirements. In this dissertation we focus on three round HAVAL for all possible output lengths. This is the first time any cryptographic analysis of HAVAL has been made public. The analysis presented in this chapter may also be applied to 4 and 5 round HAVAL.

10.3 NOTATION

Before proceeding with a description of HAVAL and the cryptanalysis of HAVAL it is appropriate to introduce the notation to be used in this chapter. The following operators are

used in this chapter:

$$\begin{aligned} \vee &= \text{Bitwise OR.} \\ \oplus &= \text{Bitwise Exclusive OR (XOR).} \\ \overline{X} &= \text{Bitwise Complement of } X \\ X^{\ggg Y} &= \text{Bitwise rotation to the right of } X \text{ by } Y \text{ positions.} \\ \wedge &= \text{Bitwise AND} \end{aligned}$$

In this chapter the bitwise AND between two variables are often indicated by x_1x_2 rather than $x_1 \wedge x_2$ for brevity. The notation of $ordi()$ for $i = 1, 2, 3, 4, 5$ indicates the word processing order for round i of the round function. The constants used by the HAVAL hash function are indicated by $K_{j,i}$ with $j = 2, 3, 4, 5$ and $i \in \{32, 33, 34, \ldots, 160\}$. There are a total of 128 additive constants used in the round functions and a further 8 constants that define the default initial values for HAVAL. The constants can be found in [62]. The constants are defined as the first 4352 bits of π. The 136 constants are not explicitly defined in this Chapter since they play no role in the analyses presented in this Chapter.

10.4 HAVAL

In this section a short description of the HAVAL hash function is presented. For a more complete description refer to [62].

Eight steps may be identified in the definition of HAVAL.

Algorithm 10.1 *HAVAL hash algorithm*

1. *Message padding.*
2. *Initialise chaining variables.*
3. *Order words for each round in the round function.*
4. *Apply compress function.*
5. *Update variables.*
6. *Has the entire message been processed ?*
   (a) *No: Repeat from step 3.*
   (b) *Yes: Continue.*
7. *Select appropriate output bits.*
8. *Output hash value.*

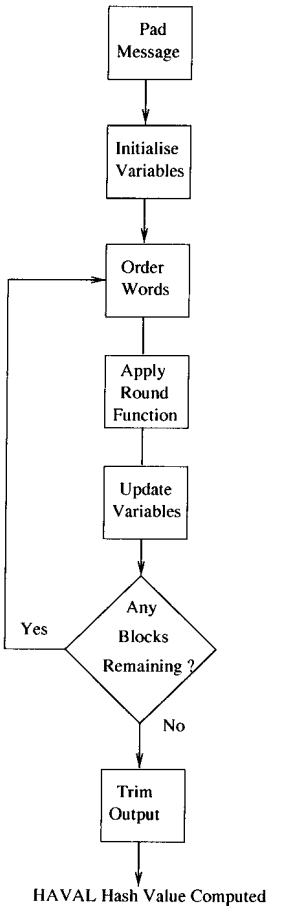

Figure 10.1: HAVAL Block Diagram

Step 1 ensures that the message is padded to a multiple of 1024 bits. Steps 3 to 5 are repeated for each 1024-bit block until the entire padded message has been processed. Step seven is used to construct the appropriate hash length (128, 160, 192, 224 or 256). Step seven is performed once the final message block is processed.

10.4.1 Message Padding

The round function of HAVAL requires a 1024-bit message block as input. Consequently the message to be hashed has to be a multiple of 1024-bits. This is accomplished by using a padding algorithm. Message padding is applied even if the unpadded message is a multiple of 1024 bits in length.

The message is padded by appending a 1 to the message, followed by m 0's ($m \leq 0$) until the length of padded message equals $944 \bmod 1024$. Once the message is padded to this length a three bit VERSION field, followed by a 3-bit PASS field and a 10-bit FPTFIELD is appended to the padded message. Once these fields are appended a 64-bit MSGLEN field is appended to form a message with a length that is a multiple of 1024. The fields mentioned above has the following meaning:

VERSION: This is a 3-bit field representing the version of HAVAL in use. The current version is 1.

PASS: This is a 3-bit field defining the number of rounds or iterations used by the round function of HAVAL. Valid values are 3, 4 or 5.

FPTFIELD: This is 10-bit field that specifies the length of the hash result. Valid values are 128, 160, 192, 224 or 256.

MSGLEN: This is a 64-bit field representing the length of the original message. The 64-bit integer is the binary representation of the length of the original message l. If l is less than 2^{32} the first 32 bits of the final 64 bits are zero.

10.4.2 Initialise Chaining Variables

The eight chaining variables are initialised to the following hexadecimal values.

$$A_0 = \texttt{0xEC4E6C89} \qquad E_0 = \texttt{0x03707344}$$
$$B_0 = \texttt{0x082EFA98} \qquad F_0 = \texttt{0x13198A2E}$$
$$C_0 = \texttt{0x299F31D0} \qquad G_0 = \texttt{0x85A308D3}$$
$$D_0 = \texttt{0xA4093822} \qquad H_0 = \texttt{0x243F6A88.}$$

10.4.3 Word Processing Order

The 1024-bit message block is divided into 32 words of 32 bits each. These words are processed in a different order for each round or iteration of the round function. The word

processing order for HAVAL is shown in Table 10.1.

ord1()	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ord2()	5	14	26	18	11	28	7	16	0	23	20	22	1	10	4	8
	30	3	21	9	17	24	29	6	19	12	15	13	2	25	31	27
ord3()	19	9	4	20	28	17	8	22	29	14	25	12	24	30	16	26
	31	15	7	3	1	0	18	27	13	6	21	10	23	11	5	2
ord4()	24	4	0	14	2	7	28	23	26	6	30	20	18	25	19	3
	22	11	31	21	8	27	12	9	1	29	5	15	17	10	16	13
ord5()	27	3	21	26	17	11	20	29	19	0	12	7	13	8	31	10
	5	9	14	30	18	6	28	24	2	23	16	22	4	1	25	15

Table 10.1: HAVAL Word Processing Order

10.4.4 Compress Function

The compress or round function is discussed in this section. Each round in the compress function utilise a different boolean function and a different word order. Each boolean function takes seven bits as input and produces a single bit as output. The boolean functions are expanded to form boolean mappings by operating bitwise on 32-bit words. The boolean mappings used in HAVAL are defined by equations (10.1) to (10.5).

$$F1(B_i, C_i, D_i, E_i, F_i, G_i, H_i) = G_iD_i \oplus F_iC_i \oplus E_iB_i \oplus H_iG_i \oplus H_i. \quad (10.1)$$

$$F2(B_i, C_i, D_i, E_i, F_i, G_i, H_i) = G_iF_iE_i \oplus C_iD_iF_i \oplus G_iF_i \oplus G_iD_i \oplus F_iB_i \oplus C_iE_i \oplus C_iD_i \oplus H_iF_i \oplus H_i. \quad (10.2)$$

$$F3(B_i, C_i, D_i, E_i, F_i, G_i, H_i) = G_iF_iE_i \oplus G_iD_i \oplus F_iC_i \oplus E_iB_i \oplus E_iH_i \oplus H_i \quad (10.3)$$

$$F4(B_i, C_i, D_i, E_i, F_i, G_i, H_i) = G_iF_iE_i \oplus F_iD_iC_i \oplus E_iD_iB_i \oplus G_iD_i \oplus F_iB_i \oplus E_iD_i \oplus E_iC_i \oplus E_iB_i \oplus D_iC_i \oplus D_iB_i \oplus H_iD_i \oplus H_i. \quad (10.4)$$

$$F5(B_i, C_i, D_i, E_i, F_i, G_i, H_i) = G_iD_i \oplus F_iC_i \oplus E_iB_i \oplus H_iG_iF_iE_i \oplus H_iC_i \oplus H_i. \quad (10.5)$$

Round 1

The first round of the HAVAL hash function is repeating the following steps for $0 \leqslant i \leqslant 31$.

$$
\begin{aligned}
1)P &= \begin{cases} F1 \circ \phi_{3,1}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=3} \\ F1 \circ \phi_{4,1}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=4} \\ F1 \circ \phi_{5,1}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=5} \end{cases} \\
2)R &= P^{\ggg 7} + A_i^{\ggg 11} + W_{ord1(i)}. \\
3)A_{i+1} &= B_i, \quad B_{i+1} = C_i, \quad C_{i+1} = D_i, \quad D_{i+1} = E_i. \\
E_{i+1} &= F_i, \quad F_{i+1} = G_i, \quad G_{i+1} = H_i, \quad H_{i+1} = R.
\end{aligned}
$$

Note that the permutation defined by $\phi_{j,1}$, $j \in \{3, 4, 5\}$ is performed before the function $F1$ is executed. The permutations are defined in Table 10.2.

Permutations	B	C	D	E	F	G	H
	↓	↓	↓	↓	↓	↓	↓
$\phi_{3,1}$	G	H	E	C	B	F	D
$\phi_{3,2}$	D	D	G	H	C	E	B
$\phi_{3,3}$	B	G	F	E	D	C	H
$\phi_{4,1}$	F	B	G	D	C	E	H
$\phi_{4,2}$	E	C	F	H	G	B	D
$\phi_{4,3}$	G	D	E	B	H	F	C
$\phi_{4,4}$	B	D	H	C	F	G	C
$\phi_{5,1}$	E	D	G	H	C	F	B
$\phi_{5,2}$	B	F	G	H	E	D	C
$\phi_{5,3}$	F	B	H	D	E	G	C
$\phi_{5,4}$	G	C	E	F	H	D	B
$\phi_{5,5}$	F	C	H	B	D	E	G

Table 10.2: Permutations (Note "↓" indicates "replace by")

Round 2

The second round of the HAVAL hash function is repeating the following steps for $32 \leqslant i \leqslant 63$.

$$
\begin{aligned}
1)P &= \begin{cases} F2 \circ \phi_{3,2}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=3} \\ F2 \circ \phi_{4,2}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=4} \\ F2 \circ \phi_{5,2}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=5} \end{cases} \\
2)R &= P^{\ggg 7} + A_i^{\ggg 11} + W_{ord2(i)} + K_{2,i}. \\
3)A_{i+1} &= B, \quad B_{i+1} = C_i, \quad C_{i+1} = D_i, \quad D_{i+1} = E_i. \\
E_{i+1} &= F_i, \quad F_{i+1} = G_i, \quad G_{i+1} = H_i, \quad H_{i+1} = R.
\end{aligned}
$$

Note that the permutation defined by $\phi_{j,2}$, $j \in \{3, 4, 5\}$ is performed before the function $F2$ is executed. The permutations are defined in Table 10.2.

Round 3

The third round of the HAVAL hash function is repeating the following steps for $64 \leqslant i \leqslant 95$.

$$
\begin{aligned}
1)P &= \begin{cases} F3 \circ \phi_{3,3}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=3} \\ F3 \circ \phi_{4,3}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=4} \\ F3 \circ \phi_{5,3}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=5} \end{cases} \\
2)R &= P^{\ggg 7} + A^{\ggg 11} + W_{ord3(i)} + K_{3,i}. \\
3)A &= B, \quad B = C, \quad C = D, \quad D = E. \\
E &= F, \quad F = G, \quad G = H, \quad H = R.
\end{aligned}
$$

Note that the permutation defined by $\phi_{j,3}$, $j \in \{3, 4, 5\}$ is performed before the function $F3$ is executed. The permutations are defined in Table 10.2.

Round 4

The fourth round of the HAVAL hash function is repeating the following steps for $96 \leqslant i \leqslant 127$.

$$\begin{aligned} 1)P &= \begin{cases} F4 \circ \phi_{4,4}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=4} \\ F4 \circ \phi_{5,4}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=5} \end{cases} \\ 2)R &= P^{\ggg 7} + A^{\ggg 11} + W_{ord4(i)} + K_{4,i}. \\ 3)A &= B, \quad B = C, \quad C = D, \quad D = E. \\ E &= F, \quad F = G, \quad G = H, \quad H = R. \end{aligned}$$

Note that the permutation defined by $\phi_{j,4}$, $j \in \{4, 5\}$ is performed before the function $F4$ is executed. The permutations are defined in Table 10.2.

Round 5

The 01 round of the HAVAL hash function is repeating the following steps for $128 \leqslant i \leqslant 159$.

$$\begin{aligned} 1)P &= \begin{cases} F5 \circ \phi_{5,5}(B_i, C_i, D_i, E_i, F_i, G_i, H_i) & \text{if PASS=5} \end{cases} \\ 2)R &= P^{\ggg 7} + A^{\ggg 11} + W_{ord5(i)} + K_{5,i}. \\ 3)A &= B, \quad B = C, \quad C = D, \quad D = E. \\ E &= F, \quad F = G, \quad G = H, \quad H = R. \end{aligned}$$

Note that the permutation defined by $\phi_{5,5}$ is performed before the function $F5$ is executed. The permutations are defined in Table 10.2.

A single step in a round of the round function is graphically represented as shown in Figure 10.2.

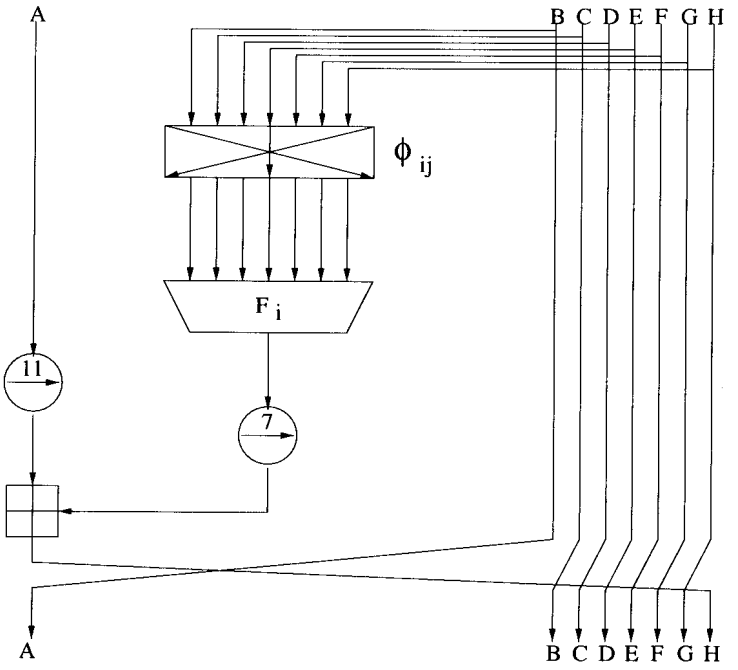

Figure 10.2: Single Step in HAVAL Hash Function

10.4.5 Tailoring the output

HAVAL can be used to produce a hash length of 128, 160, 192, 224 or 256 bits, depending on the security requirement. In this dissertation only the case where the output is 256-bits (maximum security) is considered. For a detailed description of the procedure used to select an output of less than 256 bits refer to [62]. Note that a collision for the 256-bit output implies a collision for all subsets of the output.

10.5 ANALYSIS OF HAVAL

In this section the principles derived in Chapter 8 are applied to HAVAL. In order to demonstrate the applicability of the attack it is shown that the last two rounds of three round HAVAL is not collision resistant. The last two rounds of three round HAVAL are described by steps 32 to 95. The equations describing the second round of three round HAVAL is described by

equation (10.6) for $i \in \{32, 33, 34, \ldots 63\}$.

$$\begin{aligned} H_{i+1} &= (F2 \circ \phi_{3,2}(B_i, C_i, D_i, E_i, F_i, G_i, H_i))^{\ggg 7} + A_i^{\ggg 11} + W_{ord2(i)} + K_{2,i}. \\ A_{i+1} &= B_i, \quad B_{i+1} = C_i, \quad C_{i+1} = D_i, \quad D_{i+1} = E_i \\ E_{i+1} &= F_i, \quad F_{i+1} = G_i, \quad G_{i+1} = H_i. \end{aligned} \tag{10.6}$$

The equations describing the last round of three round HAVAL is described by equation (10.7) for $i \in \{64, 65, 66, \ldots 95\}$.

$$\begin{aligned} H_{i+1} &= (F3 \circ \phi_{3,3}(B_i, C_i, D_i, E_i, F_i, G_i, H_i))^{\ggg 7} + A_i^{\ggg 11} + W_{ord3(i)} + K_{3,i}. \\ A_{i+1} &= B_i, \quad B_{i+1} = C_i, \quad C_{i+1} = D_i, \quad D_{i+1} = E_i \\ E_{i+1} &= F_i, \quad F_{i+1} = G_i, \quad G_{i+1} = H_i. \end{aligned} \tag{10.7}$$

10.5.1 Difference Equations

In order to establish a collision for the last two rounds of three round HAVAL an inner collision has to be established as described in Chapter 8. It is possible to derive a set of difference equations that allows a collision for the full 256-bit output of the last two round of three round HAVAL. One approach which results in an inner collision for the last two rounds of HAVAL is described below.

Consider steps 56 to 64 of the HAVAL as represented by equations (10.8) to (10.16).

$$\begin{aligned}
H_{57} &= (F2 \circ \phi_{3,2}(B_{56}, C_{56}, D_{56}, E_{56}, F_{56}, G_{56}, H_{56}))^{\ggg 7} + A_{56}^{\ggg 11} + W_{ord2(56)} + K_{2,56}. \\
A_{57} &= B_{56}, \quad B_{57} = C_{56}, \quad C_{57} = D_{56}, \quad D_{57} = E_{56} \\
E_{57} &= F_{56}, \quad F_{57} = G_{56}, \quad G_{57} = H_{56}. \qquad (10.8) \\
H_{58} &= (F2 \circ \phi_{3,2}(B_{57}, C_{57}, D_{57}, E_{57}, F_{57}, G_{57}, H_{57}))^{\ggg 7} + A_{57}^{\ggg 11} + W_{ord2(57)} + K_{2,57}. \\
A_{58} &= B_{57}, \quad B_{58} = C_{57}, \quad C_{58} = D_{57}, \quad D_{58} = E_{57} \\
E_{58} &= F_{57}, \quad F_{58} = G_{57}, \quad G_{58} = H_{57}. \qquad (10.9) \\
H_{59} &= (F2 \circ \phi_{3,2}(B_{58}, C_{58}, D_{58}, E_{58}, F_{58}, G_{58}, H_{58}))^{\ggg 7} + A_{58}^{\ggg 11} + W_{ord2(58)} + K_{2,58}. \\
A_{59} &= B_{58}, \quad B_{59} = C_{58}, \quad C_{59} = D_{58}, \quad D_{59} = E_{58} \\
E_{59} &= F_{58}, \quad F_{59} = G_{58}, \quad G_{59} = H_{58}. \qquad (10.10) \\
H_{60} &= (F2 \circ \phi_{3,2}(B_{59}, C_{59}, D_{59}, E_{59}, F_{59}, G_{59}, H_{59}))^{\ggg 7} + A_{59}^{\ggg 11} + W_{ord2(59)} + K_{2,59}. \\
A_{60} &= B_{59}, \quad B_{60} = C_{59}, \quad C_{60} = D_{59}, \quad D_{60} = E_{59} \\
E_{60} &= F_{59}, \quad F_{60} = G_{59}, \quad G_{60} = H_{59}. \qquad (10.11) \\
H_{61} &= (F2 \circ \phi_{3,2}(B_{60}, C_{60}, D_{60}, E_{60}, F_{60}, G_{60}, H_{60}))^{\ggg 7} + A_{60}^{\ggg 11} + W_{ord2(60)} + K_{2,60}. \\
A_{61} &= B_{60}, \quad B_{61} = C_{60}, \quad C_{61} = D_{60}, \quad D_{61} = E_{60} \\
E_{61} &= F_{60}, \quad F_{61} = G_{60}, \quad G_{61} = H_{60}. \qquad (10.12) \\
H_{62} &= (F2 \circ \phi_{3,2}(B_{61}, C_{61}, D_{61}, E_{61}, F_{61}, G_{61}, H_{61}))^{\ggg 7} + A_{61}^{\ggg 11} + W_{ord2(61)} + K_{2,61}. \\
A_{62} &= B_{61}, \quad B_{62} = C_{61}, \quad C_{62} = D_{61}, \quad D_{62} = E_{61} \\
E_{62} &= F_{61}, \quad F_{62} = G_{61}, \quad G_{62} = H_{61}. \qquad (10.13) \\
H_{63} &= (F2 \circ \phi_{3,2}(B_{62}, C_{62}, D_{62}, E_{62}, F_{62}, G_{62}, H_{62}))^{\ggg 7} + A_{62}^{\ggg 11} + W_{ord2(62)} + K_{2,62}. \\
A_{63} &= B_{62}, \quad B_{63} = C_{62}, \quad C_{63} = D_{62}, \quad D_{63} = E_{62} \\
E_{63} &= F_{62}, \quad F_{63} = G_{62}, \quad G_{63} = H_{62}. \qquad (10.14) \\
H_{64} &= (F2 \circ \phi_{3,2}(B_{63}, C_{63}, D_{63}, E_{63}, F_{63}, G_{63}, H_{63}))^{\ggg 7} + A_{63}^{\ggg 11} + W_{ord2(63)} + K_{2,63}. \\
A_{64} &= B_{63}, \quad B_{64} = C_{63}, \quad C_{64} = D_{63}, \quad D_{64} = E_{63} \\
E_{64} &= F_{63}, \quad F_{64} = G_{63}, \quad G_{64} = H_{63}. \qquad (10.15) \\
H_{65} &= (F3 \circ \phi_{3,3}(B_{64}, C_{64}, D_{64}, E_{64}, F_{64}, G_{64}, H_{64}))^{\ggg 7} + A_{64}^{\ggg 11} + W_{ord3(64)} + K_{3,64}. \\
A_{65} &= B_{64}, \quad B_{65} = C_{64}, \quad C_{65} = D_{64}, \quad D_{65} = E_{64} \\
E_{65} &= F_{64}, \quad F_{65} = G_{64}, \quad G_{65} = H_{64}. \qquad (10.16)
\end{aligned}$$

Note that $ord2(56)$ and $ord3(64)$ both select message word 19 (W_{19}). Consider two messages, M and $\tilde{M}$ that differ only in message word 19. It is then possible to derive a set of

difference equations using the principles stated in Chapter 8. In the notation of Chapter 8 we show that a message can be constructed such that:

$$f_{56}^{64}(T, M) = f_{56}^{64}(T, \tilde{M}).$$

where T represents the chaining variables $A_{56}, B_{56}, C_{56}, D_{56}, E_{56}, F_{56}, G_{56}$ and H_{56}. Once the inner collision is established it is shown that:

$$f_{32}^{95}(IV, M) = f_{32}^{95}(IV, \tilde{M}).$$

for an arbitrarily chosen IV.

The required inner collision can be established by solving the following set of difference equations

$$\begin{aligned}
H_{57} - \tilde{H}_{57} &= (F2 \circ \phi_{3,2}(B_{56}, C_{56}, D_{56}, E_{56}, F_{56}, G_{56}, H_{56}))^{\ggg 7} + A_{56}^{\ggg 11} + W_{ord2(56)} + K_{2,56} - \\
&\quad ((F2 \circ \phi_{3,2}(\tilde{B}_{56}, \tilde{C}_{56}, \tilde{D}_{56}, \tilde{E}_{56}, \tilde{F}_{56}, \tilde{G}_{56}, \tilde{H}_{56}))^{\ggg 7} + \tilde{A}_{56}^{\ggg 11} + \\
&\quad \tilde{W}_{ord2(56)} + K_{2,56}) \qquad (10.17) \\
H_{58} - \tilde{H}_{58} &= (F2 \circ \phi_{3,2}(B_{57}, C_{57}, D_{57}, E_{57}, F_{57}, G_{57}, H_{57}))^{\ggg 7} + A_{57}^{\ggg 11} + W_{ord2(57)} + K_{2,57} - \\
&\quad ((F2 \circ \phi_{3,2}(\tilde{B}_{57}, \tilde{C}_{57}, \tilde{D}_{57}, \tilde{E}_{57}, \tilde{F}_{57}, \tilde{G}_{57}, \tilde{H}_{57}))^{\ggg 7} + \tilde{A}_{57}^{\ggg 11} + \\
&\quad \tilde{W}_{ord2(57)} + K_{2,57}) \qquad (10.18) \\
H_{59} - \tilde{H}_{59} &= (F2 \circ \phi_{3,2}(B_{58}, C_{58}, D_{58}, E_{58}, F_{58}, G_{58}, H_{58}))^{\ggg 7} + A_{58}^{\ggg 11} + W_{ord2(58)} + K_{2,58} - \\
&\quad ((F2 \circ \phi_{3,2}(\tilde{B}_{58}, \tilde{C}_{58}, \tilde{D}_{58}, \tilde{E}_{58}, \tilde{F}_{58}, \tilde{G}_{58}, \tilde{H}_{58}))^{\ggg 7} + \tilde{A}_{58}^{\ggg 11} + \\
&\quad \tilde{W}_{ord2(58)} + K_{2,58}) \qquad (10.19) \\
H_{60} - \tilde{H}_{60} &= (F2 \circ \phi_{3,2}(B_{59}, C_{59}, D_{59}, E_{59}, F_{59}, G_{59}, H_{59}))^{\ggg 7} + A_{59}^{\ggg 11} + W_{ord2(59)} + K_{2,59} - \\
&\quad ((F2 \circ \phi_{3,2}(\tilde{B}_{59}, \tilde{C}_{59}, \tilde{D}_{59}, \tilde{E}_{59}, \tilde{F}_{59}, \tilde{G}_{59}, \tilde{H}_{59}))^{\ggg 7} + \tilde{A}_{59}^{\ggg 11} + \\
&\quad \tilde{W}_{ord2(59)} + K_{2,59}) \qquad (10.20)
\end{aligned}$$

$$
\begin{aligned}
H_{61} - \tilde{H}_{61} &= (F2 \circ \phi_{3,2}(B_{60}, C_{60}, D_{60}, E_{60}, F_{60}, G_{60}, H_{60}))^{\ggg 7} + A_{60}^{\ggg 11} + W_{ord2(60)} + K_{2,60} - \\
&\quad ((F2 \circ \phi_{3,2}(\tilde{B}_{60}, \tilde{C}_{60}, \tilde{D}_{60}, \tilde{E}_{60}, \tilde{F}_{60}, \tilde{G}_{60}, \tilde{H}_{60}))^{\ggg 7} + \tilde{A}_{60}^{\ggg 11} + \\
&\quad \tilde{W}_{ord2(60)} + K_{2,60}) \qquad (10.21) \\
H_{62} - \tilde{H}_{62} &= (F2 \circ \phi_{3,2}(B_{61}, C_{61}, D_{61}, E_{61}, F_{61}, G_{61}, H_{61}))^{\ggg 7} + A_{61}^{\ggg 11} + W_{ord2(61)} + K_{2,61} - \\
&\quad ((F2 \circ \phi_{3,2}(\tilde{B}_{61}, \tilde{C}_{61}, \tilde{D}_{61}, \tilde{E}_{61}, \tilde{F}_{61}, \tilde{G}_{61}, \tilde{H}_{61}))^{\ggg 7} + \tilde{A}_{61}^{\ggg 11} + \\
&\quad \tilde{W}_{ord2(61)} + K_{2,61}) \qquad (10.22) \\
H_{63} - \tilde{H}_{63} &= (F2 \circ \phi_{3,2}(B_{62}, C_{62}, D_{62}, E_{62}, F_{62}, G_{62}, H_{62}))^{\ggg 7} + A_{62}^{\ggg 11} + W_{ord2(62)} + K_{2,62} - \\
&\quad ((F2 \circ \phi_{3,2}(\tilde{B}_{62}, \tilde{C}_{62}, \tilde{D}_{62}, \tilde{E}_{62}, \tilde{F}_{62}, \tilde{G}_{62}, \tilde{H}_{62}))^{\ggg 7} + \tilde{A}_{62}^{\ggg 11} + \\
&\quad \tilde{W}_{ord2(62)} + K_{2,62}) \qquad (10.23) \\
H_{64} - \tilde{H}_{64} &= (F2 \circ \phi_{3,2}(B_{63}, C_{63}, D_{63}, E_{63}, F_{63}, G_{63}, H_{63}))^{\ggg 7} + A_{63}^{\ggg 11} + W_{ord2(63)} + K_{2,63} - \\
&\quad ((F2 \circ \phi_{3,2}(\tilde{B}_{63}, \tilde{C}_{63}, \tilde{D}_{63}, \tilde{E}_{63}, \tilde{F}_{63}, \tilde{G}_{63}, \tilde{H}_{63}))^{\ggg 7} + \tilde{A}_{63}^{\ggg 11} + \\
&\quad \tilde{W}_{ord2(63)} + K_{2,63}) \qquad (10.24) \\
H_{65} - \tilde{H}_{65} &= (F3 \circ \phi_{3,3}(B_{64}, C_{64}, D_{64}, E_{64}, F_{64}, G_{64}, H_{64}))^{\ggg 7} + A_{64}^{\ggg 11} + W_{ord3(64)} + K_{3,64} - \\
&\quad ((F3 \circ \phi_{3,3}(\tilde{B}_{64}, \tilde{C}_{64}, \tilde{D}_{64}, \tilde{E}_{64}, \tilde{F}_{64}, \tilde{G}_{64}, \tilde{H}_{64}))^{\ggg 7} + \tilde{A}_{64}^{\ggg 11} + \\
&\quad \tilde{W}_{ord3(64)} + K_{3,64}) \qquad (10.25)
\end{aligned}
$$

In order to establish an inner collision the following constraints are imposed:

$$
\begin{aligned}
H_{57} &\neq \tilde{H}_{57} \quad B_{56} = \tilde{B}_{56} \quad C_{56} = \tilde{C}_{56} \quad D_{56} = \tilde{D}_{56} \\
E_{56} &= \tilde{E}_{56} \quad F_{56} = \tilde{F}_{56} \quad G_{56} = \tilde{G}_{56} \quad H_{56} = \tilde{H}_{56} \\
W_{ord3(56)} &\neq \tilde{W}_{ord3(56)} \quad A_{56} = \tilde{A}_{56} \\
H_{65} &= \tilde{H}_{65} \quad B_{64} = \tilde{B}_{64} \quad C_{64} = \tilde{C}_{64} \quad D_{64} = \tilde{D}_{64} \\
E_{64} &= \tilde{E}_{64} \quad F_{64} = \tilde{F}_{64} \quad G_{64} = \tilde{G}_{64} \quad H_{64} = \tilde{H}_{64} \\
W_{ord3(64)} &\neq \tilde{W}_{ord3(64)}.
\end{aligned}
$$

The reader is reminded that $W_{ord3(64)} = W_{ord3(56)} = W_{19}$ and $\tilde{W}_{ord3(64)} = \tilde{W}_{ord3(56)} = \tilde{W}_{19}$.

Given these constraints the difference equations defined by (10.17) to (10.25) reduces to:

$$H_{57} - \tilde{H}_{57} = W_{19} - \tilde{W}_{19} \tag{10.26}$$

$$0 = (F2 \circ \phi_{3,2}(B_{57}, C_{57}, D_{57}, E_{57}, F_{57}, G_{57}, H_{57}))^{\ggg 7} - (F2 \circ \phi_{3,2}(B_{57}, C_{57}, D_{57}, E_{57}, F_{57}, G_{57}, \tilde{H}_{57}))^{\ggg 7} \tag{10.27}$$

$$0 = (F2 \circ \phi_{3,2}(B_{58}, C_{58}, D_{58}, E_{58}, F_{58}, G_{58}, H_{58}))^{\ggg 7} - (F2 \circ \phi_{3,2}(B_{58}, C_{58}, D_{58}, E_{58}, F_{58}, \tilde{G}_{58}, H_{58}))^{\ggg 7} \tag{10.28}$$

$$0 = (F2 \circ \phi_{3,2}(B_{59}, C_{59}, D_{59}, E_{59}, F_{59}, G_{59}, H_{59}))^{\ggg 7} - (F2 \circ \phi_{3,2}(B_{59}, C_{59}, D_{59}, E_{59}, \tilde{F}_{59}, G_{59}, H_{59}))^{\ggg 7} \tag{10.29}$$

$$0 = (F2 \circ \phi_{3,2}(B_{60}, C_{60}, D_{60}, E_{60}, F_{60}, G_{60}, H_{60}))^{\ggg 7} - (F2 \circ \phi_{3,2}(B_{60}, C_{60}, D_{60}, \tilde{E}_{60}, F_{60}, G_{60}, H_{60}))^{\ggg 7} \tag{10.30}$$

$$0 = (F2 \circ \phi_{3,2}(B_{61}, C_{61}, D_{61}, E_{61}, F_{61}, G_{61}, H_{61}))^{\ggg 7} - (F2 \circ \phi_{3,2}(B_{61}, C_{61}, \tilde{D}_{61}, E_{61}, F_{61}, G_{61}, H_{61}))^{\ggg 7}. \tag{10.31}$$

$$0 = (F2 \circ \phi_{3,2}(B_{62}, C_{62}, D_{62}, E_{62}, F_{62}, G_{62}, H_{62}))^{\ggg 7} - (F2 \circ \phi_{3,2}(B_{62}, \tilde{C}_{62}, D_{62}, E_{62}, F_{62}, G_{62}, H_{62}))^{\ggg 7}. \tag{10.32}$$

$$0 = (F2 \circ \phi_{3,2}(B_{63}, C_{63}, D_{63}, E_{63}, F_{63}, G_{63}, H_{63}))^{\ggg 7} - (F2 \circ \phi_{3,2}(\tilde{B}_{63}, C_{63}, D_{63}, E_{63}, F_{63}, G_{63}, H_{63}))^{\ggg 7}. \tag{10.33}$$

$$0 = A_{64}^{\ggg 11} + W_{19} - (\tilde{A}_{64}^{\ggg 11} + \tilde{W}_{19}) \tag{10.34}$$

From equations (10.26) to (10.34) it is observed that the chaining variables listed in Table 10.3 are affected when trying to solve the set of difference equations.

A_{56}	B_{56}	C_{56}	D_{56}	E_{56}	F_{56}	G_{56}	H_{56}	
A_{57}	B_{57}	C_{57}	D_{57}	E_{57}	F_{57}	G_{57}	H_{57}	$\tilde{H}_{57}$
A_{58}	B_{58}	C_{58}	D_{58}	E_{58}	F_{58}	G_{58}	$\tilde{G}_{58}$	H_{58}
A_{59}	B_{59}	C_{59}	D_{59}	E_{59}	F_{59}	$\tilde{F}_{59}$	G_{59}	H_{59}
A_{60}	B_{60}	C_{60}	D_{60}	E_{60}	$\tilde{E}_{60}$	F_{60}	G_{60}	H_{60}
A_{61}	B_{61}	C_{61}	D_{61}	$\tilde{D}_{61}$	E_{61}	F_{61}	G_{61}	H_{61}
A_{62}	B_{62}	C_{62}	$\tilde{C}_{62}$	D_{62}	E_{62}	F_{62}	G_{62}	H_{62}
A_{63}	B_{63}	$\tilde{B}_{63}$	C_{63}	D_{63}	E_{63}	F_{63}	G_{63}	H_{63}
A_{64}	$\tilde{B}_{64}$	B_{64}	C_{64}	D_{64}	E_{64}	F_{64}	G_{64}	H_{64}

Table 10.3: Affected Chaining Variables

Table 10.3 will be used to indicate which of the chaining variables are affected in each step of the solution of the set of difference equations. Once a chaining variable is chosen to have certain value it is marked in gray.

10.5.2 Solution to Differential Equations

In this section one technique that allows a solution to the set of differential equations described by equations (10.26) to (10.34) is described. In order to solve the set of differential equations the following properties of Boolean algebra are used [63].

For two boolean variables, denoted by x_1 and x_2, where $x_1 \neq x_2$, the following expressions hold:

$$x_1 \oplus x_1 = 0. \tag{10.35}$$

$$x_1 \wedge (x_1 \vee x_2) = x_1. \tag{10.36}$$

$$\begin{aligned} x_1 \wedge \overline{(x_1 \vee x_2)} &= x_1 \wedge \overline{x_1} \wedge \overline{x_2} \\ &= 0. \end{aligned} \tag{10.37}$$

These expressions aid in the solution of the set of differential equations defined by the next 8 steps.

Step 1

An inner collision can be established by finding a solution to equations (10.26) to (10.34). In order to find a solution to this set of difference equations it is useful to remember that $A_{64} = B_{63} = C_{62} = D_{61} = E_{60} = F_{59} = G_{58} = H_{57}$ and $\tilde{A}_{64} = \tilde{B}_{63} = \tilde{C}_{62} = \tilde{D}_{61} = \tilde{E}_{60} = \tilde{F}_{59} = \tilde{G}_{58} = \tilde{H}_{57}$. By taking the above relationships into account the following condition has to be satisfied in order to solve the set of equations.

$$H_{57} - \tilde{H}_{57} = W_{19} - \tilde{W}_{19} \tag{10.38}$$

$$H_{57}^{\ggg 11} - \tilde{A}_{57}^{\ggg 11} = \tilde{W}_{19} - W_{19} \tag{10.39}$$

This relationship can be simplified and is re-stated as:

$$\tilde{A}_{57}^{\ggg 11} - H_{57}^{\ggg 11} = H_{57} - \tilde{H}_{57}. \tag{10.40}$$

Let the difference between $H_{57} - \tilde{H}_{57}$ and $W_{19} - \tilde{W}_{19}$ be denoted by:

$$\Delta H_{57} = H_{57} - \tilde{H}_{57}. \tag{10.41}$$

$$\Delta W_{19} = W_{19} - \tilde{W}_{19}. \tag{10.42}$$

$$\Delta W_{19} = \Delta H_{57}. \tag{10.43}$$

In order for equation (10.40) to be satisfied, ΔH_{57} can be chosen as `0xAAAAAAAAA`, `0xAAAAAAAAB`, `0x55555555` or `0x55555556`. The probability that equations (10.38) and (10.39) holds for each of the possible values of ΔH_{57} are given in Table 10.4.

ΔH_{57}	Pr
0xAAAAAAAAA	0.112
0xAAAAAAAAB	0.448
0x555555555	0.448
0x555555556	0.110

Table 10.4: Probability that a given ΔH satisfies (10.38) and (10.39) for random values of H_{57}

Here H_{57} can be selected at random. Note that any choice for H_{57} implies a selection for $\tilde{H}_{57}$ through the chosen relationship of ΔH_{57}. If equation (10.38) and (10.39) holds for the chosen H_{57}, continue with Step 2. The affected chaining variables for Step 1 are shown in Table 10.5.

A_{56}	B_{56}	C_{56}	D_{56}	E_{56}	F_{56}	G_{56}	H_{56}	
A_{57}	B_{57}	C_{57}	D_{57}	E_{57}	F_{57}	G_{57}	H_{57}	$\tilde{H}_{57}$
A_{58}	B_{58}	C_{58}	D_{58}	E_{58}	F_{58}	G_{58}	$\tilde{G}_{58}$	H_{58}
A_{59}	B_{59}	C_{59}	D_{59}	E_{59}	F_{59}	$\tilde{F}_{59}$	G_{59}	H_{59}
A_{60}	B_{60}	C_{60}	D_{60}	E_{60}	$\tilde{E}_{60}$	F_{60}	G_{60}	H_{60}
A_{61}	B_{61}	C_{61}	D_{61}	$\tilde{D}_{61}$	E_{61}	F_{61}	G_{61}	H_{61}
A_{62}	B_{62}	C_{62}	$\tilde{C}_{62}$	D_{62}	E_{62}	F_{62}	G_{62}	H_{62}
A_{63}	B_{63}	$\tilde{B}_{63}$	C_{63}	D_{63}	E_{63}	F_{63}	G_{63}	H_{63}
A_{64}	$\tilde{A}_{64}$	B_{64}	C_{64}	D_{64}	E_{64}	F_{64}	G_{64}	H_{64}

Table 10.5: Affected Chaining Variables: Step 1

Step 2

Once suitable values for H_{57} and $\tilde{H}_{57}$ have been found, a solution has to be found to equation (10.33). Equation (10.33) can be solved by applying the permutation $\phi_{3,2}$ and considering the resulting function $F2$. Equation (10.33) can then be written as:

$$\begin{aligned} 0 \;=\; & (E_{63}C_{63}H_{63} \oplus F_{63}G_{63}C_{63} \oplus E_{63}C_{63} \oplus E_{63}G_{63} \oplus C_{63}D_{63} \oplus F_{63}H_{63} \oplus F_{63}G_{63} \oplus \\ & B_{63}C_{63} \oplus B_{63}) - (E_{63}C_{63}H_{63} \oplus F_{63}G_{63}C_{63} \oplus E_{63}C_{63} \oplus E_{63}G_{63} \oplus C_{63}D_{63} \oplus \\ & F_{63}H_{63} \oplus F_{63}G_{63} \oplus \tilde{B}_{63}C_{63} \oplus \tilde{B}_{63}). \end{aligned} \tag{10.44}$$

If only the terms that differ from each other are considered equation (10.44) reduces to:

$$0 \;=\; (B_{63}C_{63} \oplus B_{63}) - (\tilde{B}_{63}C_{63} \oplus \tilde{B}_{63}) \tag{10.45}$$

$$(B_{63}C_{63} \oplus B_{63}) \;=\; (\tilde{B}_{63}C_{63} \oplus \tilde{B}_{63}) \tag{10.46}$$

By setting C_{63} to:

$$C_{63} = B_{63} \vee \tilde{B}_{63}. \tag{10.47}$$

Consequently equation (10.46) reduces to:

$$(B_{63} \oplus B_{63}) \;=\; (\tilde{B}_{63} \oplus \tilde{B}_{63}) \tag{10.48}$$

$$0 \;=\; 0. \tag{10.49}$$

Thus our choice for C_{63} satisfies equation (10.46). The chaining variable table after the completion of step 2 is shown below.

A_{56}	B_{56}	C_{56}	D_{56}	E_{56}	F_{56}	G_{56}	H_{56}	
A_{57}	B_{57}	C_{57}	D_{57}	E_{57}	F_{57}	G_{57}	H_{57}	$\tilde{H}_{57}$
A_{58}	B_{58}	C_{58}	D_{58}	E_{58}	F_{58}	G_{58}	$\tilde{G}_{58}$	H_{58}
A_{59}	B_{59}	C_{59}	D_{59}	E_{59}	F_{59}	$\tilde{F}_{59}$	G_{59}	H_{59}
A_{60}	B_{60}	C_{60}	D_{60}	E_{60}	$\tilde{E}_{60}$	F_{60}	G_{60}	H_{60}
A_{61}	B_{61}	C_{61}	D_{61}	$\tilde{D}_{61}$	E_{61}	F_{61}	G_{61}	H_{61}
A_{62}	B_{62}	C_{62}	$\tilde{C}_{62}$	D_{62}	E_{62}	F_{62}	G_{62}	H_{62}
A_{63}	B_{63}	$\tilde{B}_{63}$	C_{63}	D_{63}	E_{63}	F_{63}	G_{63}	H_{63}
A_{64}	$\tilde{A}_{64}$	B_{64}	C_{64}	D_{64}	E_{64}	F_{64}	G_{64}	H_{64}

Table 10.6: Affected Chaining Variables: Step 2

Step 3

The next step is to solve equation (10.32). As before equation (10.32) can be solved by applying the permutation $\phi_{3,2}$ and considering the resulting function $F2$. Equation (10.32) can then be written as:

$$\begin{aligned} 0 &= (E_{62}C_{62}H_{62} \oplus F_{62}G_{62}C_{62} \oplus E_{62}C_{62} \oplus E_{62}G_{62} \oplus C_{62}D_{62} \oplus F_{62}H_{62} \oplus F_{62}G_{62} \oplus \\ &\quad B_{62}C_{62} \oplus B_{62}) - (E_{62}\tilde{C}_{62}H_{62} \oplus F_{62}G_{62}\tilde{C}_{62} \oplus E_{62}\tilde{C}_{62} \oplus E_{62}G_{62} \oplus \tilde{C}_{62}D_{62} \oplus \\ &\quad F_{62}H_{62} \oplus F_{62}G_{62} \oplus B_{62}\tilde{C}_{62} \oplus B_{62}). \end{aligned} \tag{10.50}$$

If only the terms that differ from each other are considered equation (10.50) reduces to:

$$\begin{aligned} 0 &= (E_{62}C_{62}H_{62} \oplus F_{62}G_{62}C_{62} \oplus E_{62}C_{62} \oplus C_{62}D_{62} \oplus B_{62}C_{62}) - \\ &\quad (E_{62}\tilde{C}_{62}H_{62} \oplus F_{62}G_{62}\tilde{C}_{62} \oplus E_{62}\tilde{C}_{62} \oplus \tilde{C}_{62}D_{62} \oplus B_{62}\tilde{C}_{62}). \end{aligned} \tag{10.51}$$

Equation (10.51) holds if the following expressions hold true:

$$E_{62}C_{62}H_{62} = E_{62}\tilde{C}_{62}H_{62}. \tag{10.52}$$

$$E_{62}C_{62} = E_{62}\tilde{C}_{62}. \tag{10.53}$$

$$F_{62}G_{62}C_{62} \oplus C_{62}D_{62} = \oplus F_{62}G_{62}\tilde{C}_{62} \oplus \tilde{C}_{62}D_{62}. \tag{10.54}$$

$$B_{62}C_{62} = B_{62}\tilde{C}_{62}. \tag{10.55}$$

Chaining variable B_{62} and E_{62} is chosen such that:

$$B_{62} = \overline{C_{62} \vee \tilde{C}_{62}}. \tag{10.56}$$

$$E_{62} = \overline{C_{62} \vee \tilde{C}_{62}}. \tag{10.57}$$

This choice of B_{62} reduces equation (10.55) to:

$$B_{62}C_{62} = B_{62}\tilde{C}_{62} \tag{10.58}$$

$$\overline{C}_{62}C_{62}\tilde{C}_{62} = \overline{\tilde{C}}_{62}\tilde{C}_{62}C_{62} \tag{10.59}$$

$$0 = 0. \tag{10.60}$$

Likewise equation (10.53) reduces to:

$$E_{62}C_{62} = E_{62}\tilde{C}_{62} \tag{10.61}$$

$$\overline{C}_{62}C_{62}\tilde{C}_{62} = \overline{\tilde{C}}_{62}\tilde{C}_{62}C_{62} \tag{10.62}$$

$$0 = 0. \tag{10.63}$$

Similarly equation (10.53) reduces to:

$$E_{62}C_{62}H_{62} = E_{62}\tilde{C}_{62}H_{62}. \quad (10.64)$$

$$\overline{C}_{62}C_{62}\tilde{C}_{62}H_{62} = \overline{\tilde{C}}_{62}\tilde{C}_{62}C_{62}H_{62}. \quad (10.65)$$

$$0 \cdot H_{62} = 0 \cdot H_{62}. \quad (10.66)$$

$$0 = 0. \quad (10.67)$$

Note that no explicit choice is made for H_{62}. Chaining variable D_{62} is already determined through the choice for C_{63}. By setting:

$$G_{62} = C_{62} \vee \tilde{C}_{62}. \quad (10.68)$$

$$F_{62} = C_{62} \vee \tilde{C}_{62}. \quad (10.69)$$

It is assured that equation (10.54) holds.

The chaining variable table after the completion of step 3 is shown in Table 10.7.

A_{56}	B_{56}	C_{56}	D_{56}	E_{56}	F_{56}	G_{56}	H_{56}	
A_{57}	B_{57}	C_{57}	D_{57}	E_{57}	F_{57}	G_{57}	H_{57}	$\tilde{H}_{57}$
A_{58}	B_{58}	C_{58}	D_{58}	E_{58}	F_{58}	G_{58}	$\tilde{G}_{58}$	H_{58}
A_{59}	B_{59}	C_{59}	D_{59}	E_{59}	F_{59}	$\tilde{F}_{59}$	G_{59}	H_{59}
A_{60}	B_{60}	C_{60}	D_{60}	E_{60}	$\tilde{E}_{60}$	F_{60}	G_{60}	H_{60}
A_{61}	B_{61}	C_{61}	D_{61}	$\tilde{D}_{61}$	E_{61}	F_{61}	G_{61}	H_{61}
A_{62}	B_{62}	C_{62}	$\tilde{C}_{62}$	D_{62}	E_{62}	F_{62}	G_{62}	H_{62}
A_{63}	B_{63}	$\tilde{B}_{63}$	C_{63}	D_{63}	E_{63}	F_{63}	G_{63}	H_{63}
A_{64}	$\tilde{A}_{64}$	B_{64}	C_{64}	D_{64}	E_{64}	F_{64}	G_{64}	H_{64}

Table 10.7: Affected Chaining Variables: Step 3

Step 4

In this step equation (10.31) is solved. Equation (10.31) can be solved by applying the permutation $\phi_{3,2}$ and considering the resulting function $F2$. Equation (10.31) then reduces

to:

$$\begin{aligned} 0 &= (E_{61}C_{61}H_{61} \oplus F_{61}G_{61}C_{61} \oplus E_{61}C_{61} \oplus E_{61}G_{61} \oplus C_{61}D_{61} \oplus F_{61}H_{61} \oplus \\ &\quad F_{61}G_{61} \oplus B_{61}C_{61} \oplus B_{61}) - (E_{61}C_{61}H_{61} \oplus F_{61}G_{61}C_{61} \oplus E_{61}C_{61} \oplus \\ &\quad E_{61}G_{61} \oplus C_{61}\tilde{D}_{61} \oplus F_{61}H_{61} \oplus F_{61}G_{61} \oplus B_{61}C_{61} \oplus B_{61}). \end{aligned} \tag{10.70}$$

Consider only the terms that differ from each other. Then equation (10.70) reduces to:

$$0 = (C_{61}D_{61}) - (C_{61}\tilde{D}_{61}). \tag{10.71}$$

The choice for B_{62} in Step 3 also ensures that equation (10.70) holds. Consequently the chaining variable table remains unchanged (see Table 10.7) after the completion of Step 4.

Step 5

In this step equation (10.30) is solved. Equation (10.30) can be solved by applying the permutation $\phi_{3,2}$ and considering the resulting function expression. Equation (10.30) can then be written as:

$$\begin{aligned} 0 &= (E_{60}C_{60}H_{60} \oplus F_{60}G_{60}C_{60} \oplus E_{60}C_{60} \oplus E_{60}G_{60} \oplus C_{60}D_{60} \oplus F_{60}H_{60} \oplus \\ &\quad F_{60}G_{60} \oplus B_{60}C_{60} \oplus B_{60}) - (\tilde{E}_{60}C_{60}H_{60} \oplus F_{60}G_{60}C_{60} \oplus \tilde{E}_{60}C_{60} \oplus \\ &\quad \tilde{E}_{60}G_{60} \oplus C_{60}D_{60} \oplus F_{60}H_{60} \oplus F_{60}G_{60} \oplus B_{60}C_{60} \oplus B_{60}). \end{aligned} \tag{10.72}$$

If only the terms that differ from each other are considered equation (10.72) reduces to:

$$0 = (E_{60}C_{60}H_{60} \oplus E_{60}C_{60} \oplus E_{60}G_{60}) - (\tilde{E}_{60}C_{60}H_{60} \oplus \tilde{E}_{60}C_{60} \oplus \tilde{E}_{60}G_{60}) \tag{10.73}$$

Equation (10.73) holds if the following expressions hold true:

$$E_{60}C_{60}H_{60} = \tilde{E}_{60}C_{60}H_{60}. \tag{10.74}$$

$$E_{60}C_{60} = \tilde{E}_{60}C_{60}. \tag{10.75}$$

$$E_{60}G_{60} = \tilde{E}_{60}G_{60}. \tag{10.76}$$

$$\tag{10.77}$$

The value for G_{60} is already determined by the choice made for E_{62} in Step 3. The value chosen in Step 3 for E_{62} satisfies equation (10.76). Similarly the previous choice for F_{62} implies that the value for H_{60} is fixed. Likewise it is found that this particular choice for F_{62} assures that equation (10.74) holds if C_{60} is chosen such that:

$$C_{60} = \overline{E_{60} \vee \tilde{E}_{60}}. \tag{10.78}$$

This choice for C_{60} also assures that equation (10.75) holds. The chaining variable table can now be updated to reflect the additional choices made. Table 10.8 is shown below.

A_{56}	B_{56}	C_{56}	D_{56}	E_{56}	F_{56}	G_{56}	H_{56}	
A_{57}	B_{57}	C_{57}	D_{57}	E_{57}	F_{57}	G_{57}	H_{57}	$\tilde{H}_{57}$
A_{58}	B_{58}	C_{58}	D_{58}	E_{58}	F_{58}	G_{58}	$\tilde{G}_{58}$	H_{58}
A_{59}	B_{59}	C_{59}	D_{59}	E_{59}	F_{59}	$\tilde{F}_{59}$	G_{59}	H_{59}
A_{60}	B_{60}	C_{60}	D_{60}	E_{60}	$\tilde{E}_{60}$	F_{60}	G_{60}	H_{60}
A_{61}	B_{61}	C_{61}	D_{61}	$\tilde{D}_{61}$	E_{61}	F_{61}	G_{61}	H_{61}
A_{62}	B_{62}	C_{62}	$\tilde{C}_{62}$	D_{62}	E_{62}	F_{62}	G_{62}	H_{62}
A_{63}	B_{63}	$\tilde{B}_{63}$	C_{63}	D_{63}	E_{63}	F_{63}	G_{63}	H_{63}
A_{64}	$\tilde{A}_{64}$	B_{64}	C_{64}	D_{64}	E_{64}	F_{64}	G_{64}	H_{64}

Table 10.8: Affected Chaining Variables: Step 5

Step 6

In this step equation (10.29) is solved. Equation (10.29) can be solved by applying the permutation $\phi_{3,2}$ and considering the resulting function expression. Equation (10.29) can then be written as:

$$\begin{aligned} 0 \;=\; & (E_{59}C_{59}H_{59} \oplus F_{59}G_{59}C_{59} \oplus E_{59}C_{59} \oplus E_{59}G_{59} \oplus C_{59}D_{59} \oplus F_{59}H_{59} \oplus \\ & F_{59}G_{59} \oplus B_{59}C_{59} \oplus B_{59}) - (E_{59}C_{59}H_{59} \oplus \tilde{F}_{59}G_{59}C_{59} \oplus E_{59}C_{59} \oplus \\ & E_{59}G_{59} \oplus C_{59}D_{59} \oplus \tilde{F}_{59}H_{59} \oplus \tilde{F}_{59}G_{59} \oplus B_{59}C_{59} \oplus B_{59}). \end{aligned} \tag{10.79}$$

If only the terms that differ from each other are considered equation (10.79) reduces to:

$$0 \;=\; (F_{59}G_{59}C_{59} \oplus F_{59}H_{59} \oplus F_{59}G_{59}) - (\tilde{F}_{59}G_{59}C_{59} \oplus \tilde{F}_{59}H_{59} \oplus \tilde{F}_{59}G_{59}) \tag{10.80}$$

Equation (10.80) holds if the following expressions hold true:

$$F_{59}G_{59}C_{59} \oplus F_{59}G_{59} = \tilde{F}_{59}G_{59}C_{59} \oplus \tilde{F}_{59}G_{59}. \quad (10.81)$$

$$F_{59}H_{59} = \tilde{F}_{59}H_{59}. \quad (10.82)$$

$$(10.83)$$

The values for G_{59} and H_{59} are already determined by the choices made for D_{62} and E_{62} in Step 3. The values chosen in Step 3 for D_{62} and E_{60} also satisfies equations (10.81) and (10.82) C_{59} is chosen such that:

$$C_{59} = F_{59} \vee \tilde{F}_{59}. \quad (10.84)$$

The updated chaining variable table is presented as Table 10.9.

A_{56}	B_{56}	C_{56}	D_{56}	E_{56}	F_{56}	G_{56}	H_{56}	
A_{57}	B_{57}	C_{57}	D_{57}	E_{57}	F_{57}	G_{57}	H_{57}	$\tilde{H}_{57}$
A_{58}	B_{58}	C_{58}	D_{58}	E_{58}	F_{58}	G_{58}	$\tilde{G}_{58}$	H_{58}
A_{59}	B_{59}	C_{59}	D_{59}	E_{59}	F_{59}	$\tilde{F}_{59}$	G_{59}	H_{59}
A_{60}	B_{60}	C_{60}	D_{60}	E_{60}	$\tilde{E}_{60}$	F_{60}	G_{60}	H_{60}
A_{61}	B_{61}	C_{61}	D_{61}	$\tilde{D}_{61}$	E_{61}	F_{61}	G_{61}	H_{61}
A_{62}	B_{62}	C_{62}	$\tilde{C}_{62}$	D_{62}	E_{62}	F_{62}	G_{62}	H_{62}
A_{63}	B_{63}	$\tilde{B}_{63}$	C_{63}	D_{63}	E_{63}	F_{63}	G_{63}	H_{63}
A_{64}	$\tilde{A}_{64}$	B_{64}	C_{64}	D_{64}	E_{64}	F_{64}	G_{64}	H_{64}

Table 10.9: Affected Chaining Variables: Step 6

Step 7

In this step equation (10.28) is solved. As before consider the expression obtained by applying the permutation $\phi_{3,2}$ to equation (10.28) and considering the resulting function expression. Equation (10.28) can then be written as:

$$\begin{aligned} 0 = {} & (E_{58}C_{58}H_{58} \oplus F_{58}G_{58}C_{58} \oplus E_{58}C_{58} \oplus E_{58}G_{58} \oplus C_{58}D_{58} \oplus F_{58}H_{58} \oplus - \\ & F_{58}G_{58} \oplus B_{58}C_{58} \oplus B_{58})(E_{58}C_{58}H_{58} \oplus F_{58}\tilde{G}_{58}C_{58} \oplus E_{58}C_{58} \oplus \\ & E_{58}\tilde{G}_{58} \oplus C_{58}D_{58} \oplus F_{58}H_{58} \oplus F_{58}\tilde{G}_{58} \oplus B_{58}C_{58} \oplus B_{58}). \end{aligned} \quad (10.85)$$

consider only the terms that differ from each other. Then equation (10.85) reduces to:

$$\begin{aligned} 0 &= (F_{58}G_{58}C_{58} \oplus E_{58}G_{58} \oplus F_{58}G_{58}) - \\ &\quad (F_{58}\tilde{G}_{58}C_{58} \oplus E_{58}\tilde{G}_{58} \oplus F_{58}\tilde{G}_{58}). \end{aligned} \tag{10.86}$$

Equation (10.86) holds if the following expressions hold true:

$$F_{58}G_{58}C_{58} = F_{58}\tilde{G}_{58}C_{58}. \tag{10.87}$$

$$E_{58}G_{58} = E_{58}\tilde{G}_{58}. \tag{10.88}$$

$$F_{58}G_{58} = F_{58}\tilde{G}_{58}. \tag{10.89}$$

$$\tag{10.90}$$

The value for F_{58} is determined by the choice for B_{62} in Step 5. This choice for F_{58} also ensures that equations (10.87) and (10.89) holds. The value for E_{58} is determined by the value chosen for C_{60} in step 5. This choice also assures that equation (10.88) holds. The chaining variable table remains unchanged after the completion of step 7.

Step 8

In this step equation (10.27) is solved. As before consider the expression obtained by applying the permutation $\phi_{3,2}$ to equation (10.27) and considering the resulting function expression. Equation (10.27) can then be written as:

$$\begin{aligned} 0 &= (E_{57}C_{57}H_{57} \oplus F_{57}G_{57}C_{57} \oplus E_{57}C_{57} \oplus E_{57}G_{57} \oplus C_{57}D_{57} \oplus F_{57}H_{57} \oplus \\ &\quad F_{57}G_{57} \oplus B_{57}C_{57} \oplus B_{57}) - (E_{57}C_{57}\tilde{H}_{57} \oplus F_{57}G_{57}C_{57} \oplus E_{57}C_{57} \oplus \\ &\quad E_{57}G_{57} \oplus C_{57}D_{57} \oplus F_{57}\tilde{H}_{57} \oplus F_{57}G_{57} \oplus B_{57}C_{57} \oplus B_{57}). \end{aligned} \tag{10.91}$$

consider only the terms that differ from each other. Then equation (10.91) reduces to:

$$\begin{aligned} 0 &= (E_{57}C_{57}H_{57} \oplus F_{57}H_{57}) - \\ &\quad (E_{57}C_{57}\tilde{H}_{57} \oplus F_{57}\tilde{H}_{57}). \end{aligned} \tag{10.92}$$

Equation (10.92) holds if the following expressions hold.

$$E_{57}C_{57}H_{57} = E_{57}C_{57}\tilde{H}_{57}. \tag{10.93}$$

$$F_{57}H_{57} = F_{57}\tilde{H}_{57}. \tag{10.94}$$

The value for F_{57} is determined by the choice for C_{60} in step 5. This choice also satisfies equation (10.94). Similarly the value of E_{57} is determined by the choice for C_{59} in step 6. Given this value for E_{57} equation (10.93) holds if C_{57} is chosen such that:

$$C_{57} = \overline{H_{57} \vee \tilde{H}_{57}}. \tag{10.95}$$

The updated chaining variable table is shown in Table 10.10.

A_{56}	B_{56}	C_{56}	D_{56}	E_{56}	F_{56}	G_{56}	H_{56}	
A_{57}	B_{57}	C_{57}	D_{57}	E_{57}	F_{57}	G_{57}	H_{57}	$\tilde{H}_{57}$
A_{58}	B_{58}	C_{58}	D_{58}	E_{58}	F_{58}	G_{58}	G_{58}	H_{58}
A_{59}	B_{59}	C_{59}	D_{59}	E_{59}	F_{59}	$\tilde{F}_{59}$	G_{59}	H_{59}
A_{60}	B_{60}	C_{60}	D_{60}	E_{60}	$\tilde{E}_{60}$	F_{60}	G_{60}	H_{60}
A_{61}	B_{61}	C_{61}	D_{61}	$\tilde{D}_{61}$	E_{61}	F_{61}	G_{61}	H_{61}
A_{62}	B_{62}	C_{62}	$\tilde{C}_{62}$	D_{62}	E_{62}	F_{62}	G_{62}	H_{62}
A_{63}	B_{63}	$\tilde{B}_{63}$	C_{63}	D_{63}	E_{63}	F_{63}	G_{63}	H_{63}
A_{64}	$\tilde{A}_{64}$	B_{64}	C_{64}	D_{64}	E_{64}	F_{64}	G_{64}	H_{64}

Table 10.10: Affected Chaining Variables: Step 8

After the completion of step 8 the set of difference equations defined by equations (10.26) to (10.34) are solved. The chaining variables not marked in gray can take on randomly selected values.

Message Construction

Once the set of difference equations is solved it remains to construct the two messages that will result in a collision for the last two rounds of three round HAVAL. In general a message word for the second round can be derived using the following equation:

$$W_{ord2(i)} = H_{i+1} - (F2 \circ \phi_{3,2}(B_i, C_i, D_i, E_i, F_i, G_i, H_i))^{\gg 7} - A_i^{\gg 11} - K_{2,i}. \tag{10.96}$$

By applying equation (10.96) for $32 \leqslant i \leqslant 63$ two messages, M and $\tilde{M}$ can be derived. In order to meet a specific initial value ($i = 32$) appropriate selections should be made for H_{32}, H_{33}, H_{34}, H_{35}, H_{36}, H_{37}, H_{38} and H_{39}. An implementation of this attack is included in Appendix G.

10.5.3 Collision Example

A collision for the last two rounds of HAVAL was constructed using the techniques described in this chapter.

For the initial values of:

$$\begin{aligned} A_{32} &= \texttt{0xEC4E6C89} \\ B_{32} &= \texttt{0x082EFA98} \\ C_{32} &= \texttt{0x299F31D0} \\ D_{32} &= \texttt{0xA4093822} \\ E_{32} &= \texttt{0x03707344} \\ F_{32} &= \texttt{0x13198A2E} \\ G_{32} &= \texttt{0x85A308D3} \\ H_{32} &= \texttt{0x243F6A88}. \end{aligned}$$

The derived message is defined by:

$$
\begin{aligned}
W_0 &= \texttt{0x3A379ED0} & W_{16} &= \texttt{0x79F23F4E}\\
W_1 &= \texttt{0x1EB81543} & W_{17} &= \texttt{0x9C1596E8}\\
W_2 &= \texttt{0x279C0073} & W_{18} &= \texttt{0xB62B4D8B}\\
W_3 &= \texttt{0xC9295C45} & W_{19} &= \texttt{0xDEC04668}\\
W_4 &= \texttt{0x6988BCBA} & W_{20} &= \texttt{0x4BA12694}\\
W_5 &= \texttt{0xEE1E55A2} & W_{21} &= \texttt{0x9D8DED5C}\\
W_6 &= \texttt{0xDE458436} & W_{22} &= \texttt{0x456CFCB4}\\
W_7 &= \texttt{0xB1C55B3C} & W_{23} &= \texttt{0x7253D2B9}\\
W_8 &= \texttt{0xBDA229EB} & W_{24} &= \texttt{0x61ED5DB4}\\
W_9 &= \texttt{0xE27926BE} & W_{25} &= \texttt{0xE4C2E748}\\
W_{10} &= \texttt{0x8BACAC22} & W_{26} &= \texttt{0xFD80A2AD}\\
W_{11} &= \texttt{0xBDF710B4} & W_{27} &= \texttt{0xC033F56E}\\
W_{12} &= \texttt{0x6516723B} & W_{28} &= \texttt{0x3010FDA9}\\
W_{13} &= \texttt{0x26991773} & W_{29} &= \texttt{0x344A7F71}\\
W_{14} &= \texttt{0x9EA6FD1F} & W_{30} &= \texttt{0x0DB561C7}\\
W_{15} &= \texttt{0x0BB27961} & W_{31} &= \texttt{0xC7A1E175}
\end{aligned}
$$

The alternative message is identical to the first with the exception that $\tilde{W}_{19}$ is chosen such that:

$$\tilde{W}_{19} = \texttt{0x34159BBD}$$

The resulting collision (including the feed forward step) for the last two rounds of three round HAVAL is:

$$
\begin{aligned}
f_{32}^{95}(IV, W) &= \texttt{0x4DF09D997588F9C7BE20863B2EED2AAC}\\
&\quad\ \texttt{D5B1E116D927279E250D19CB00850706}.
\end{aligned}
$$

10.6 CONCLUSION

In this chapter it was shown that the generalised technique described in Chapter 8 can be applied to the HAVAL hash function. In particular it is shown that the last two rounds of three

round HAVAL is not collision resistant. It is demonstrated that a collision can be established for all 256 output bits produced. This attack is considerably more efficient than the birthday attack which would require 2^{128} evaluations of the last two rounds of HAVAL. This is the first cryptanalytical results obtained for the HAVAL hash function. It is believed that it is possible to extend the attack to all three rounds of HAVAL. The attack can be executed on a 200 MHz Pentium Pro in less than a minute. Source code that implements this attack is attached as Appendix G.

CHAPTER 11: DESIGN CRITERIA FOR DEDICATED HASH FUNCTIONS

11.1 INTRODUCTION

In this chapter guidelines and design criteria for the design of dedicated hash functions based on the MD4 family of hash functions are presented. The discussion of the design criteria in this chapter is based on the experience gained from the analysis of dedicated hash functions included in the MD4 family of hash functions. These hash functions include MD4, MD5, HAVAL, SHA and SHA-1. These hash functions share a common ancestry (MD4), and consequently they share a number of features. These hash functions also differ in a number of respects, such as the Boolean mappings and the message word re-use mechanisms.

Each of the components encountered in the MD4 family of hash functions is discussed with regard to their contribution to the security of the hash functions in which they occur. In this chapter the emphasis is on the security requirements expected from the building blocks with occasional reference to the functional requirements.

11.2 BASIC STRUCTURE

The compress function construction used for MD4 is described in [10]. This construction has been widely adopted in the design of other hash functions such as MD5 [45], SHA, SHA-1 [13] and Tiger [47]

The MD4 family of hash functions takes two parameters as inputs namely the previous hash result and the message block. If the first message block is processed, the previous hash result is replaced by the initial value. The generalised MD4 family construction does not allow for the inclusion of a secret key. A survey of a number of adaptations of this construction that does make allowance for a secret key is presented in Chapter 5.

The compress function used in the MD4 family of constructions, is an iterated construction. The compress function takes as input the previous hash result H_{i-1} and the current message block, M_i.

The compress function consists of a number of rounds. These rounds are constructed from a

number of steps. Each step is constructed from a number of elementary operations, including Boolean mappings, rotations and additions $\mathrm{mod} 2^{32}$.

The message block M_i is segmented into k sub-blocks. The initial chaining variable, C, is set equal to the previous hash result. The set of message sub-blocks is divided into a number of l-bit message words. These message words are re-used in consecutive rounds of the compress function according to a specified rule. The chaining variable, C, is updated in each step of each round of the compress function. The output of the compress function is obtained by adding the initial value of the chaining variable C (i.e. the previous output of the compress function or the initial value) to the final value of the chaining variable. The final hash value for the message is the output obtained from the final application of the compress function.

11.3 BUILDING BLOCKS

A number of basic building blocks are used in the construction of the compress functions of the MD4 family of hash functions. These include message expansion algorithms, message block permutations, rotations, addition $\mathrm{mod}\ 2^{32}$ and Boolean mappings or S-boxes. In this section the contribution of each of these basic building blocks to the security of the dedicated hash functions is considered.

11.3.1 Boolean Mappings

The Boolean mappings used in MD4, MD5, SHA, SHA-1 and HAVAL are constructed with the same technique and exhibit similar properties. The Boolean mappings used by these functions take a number of 32 bit input words and produce a single 32 bit output word.

The Boolean mappings utilised by these hash functions are constructed from Boolean functions. A number of design criteria for Boolean functions are established in Chapter 3 of [51]. These criteria deal with the zero-one balance, high non-linearity values and the propagation properties of the Boolean functions. In the definition of MD4 [10] and MD5 [45] it is stated that if the inputs to the Boolean function are independent and unbiased, then the output of the Boolean function will be independent and unbiased. The functions defined for MD4 and MD5 are used in both SHA and SHA-1. It should be noted that a number of the

Boolean functions defined for use with MD4 and MD5 do not satisfy the criteria set forth in [51]. HAVAL represents a family of hash functions derived from MD4 and is defined in [62]. The Boolean functions used in HAVAL are derived from bent functions and were designed to have zero-one balance, a high non-linearity and satisfy the strict avalanche criterion (SAC). In addition these functions are linearly inequivalent in structure and are mutually output-uncorrelated.

Having obtained a Boolean function which satisfies the desired properties, a Boolean mapping is constructed by applying the Boolean function to a number of bits in parallel. Boolean mappings constructed in this manner inherits the properties of the Boolean function. The number of bits are chosen to reflect a specific computer architecture. Currently 32 bit machines are in common use and consequently the Boolean mappings are defined over 32 bit variables. Many general purpose processors contain logical bitwise operators in their instruction sets. Thus the use of bitwise logical functions as Boolean mappings are advantageous, if the overall performance of the hash function is considered.

It is maintained that the design criteria applied to construct the Boolean mappings used by the MD4 family are necessary, but not sufficient. The practice of extending the Boolean function to a Boolean mapping by applying the Boolean function in parallel to a number of bits, has proved to be a salient point in the cryptanalysis of MD4, MD5 and HAVAL (Chapters 6 7 10). If a single input bit used by the Boolean mapping is changed, at most a single output bit of the Boolean mapping is changed. In addition, the compliance to the SAC by the Boolean function used to construct the Boolean mapping, implies that a single input bit may be changed without any changes occurring in the output of the Boolean mapping. For these reasons it is proposed that the Boolean mapping should be constructed to satisfy the bit independence criterion (BIC) as described in [51].

The only dedicated hash function which uses randomly generated Boolean mappings is Tiger [47]. This hash function was designed with 64-bit architectures in mind. For this reason it utilises four 8×64 bit S-boxes. The following design criteria for the Boolean mappings are quoted verbatim from [47].

1. All the entries of all the S boxes should be distinct. Moreover, no two entries should have more than three equal bytes.

2. Each byte-column is a permutation of all the 256 possible byte values.

3. The columns of all the S boxes should be as different as possible, and have some long cycles.

4. No two differences of S box entries ($S_i(t_1) \oplus S_i(t_2)$ and $S_j(t_3) \oplus S_j(t_4)$) should have more than four equal bytes.

5. The speed of the generation should not be too slow, in order to enable applications to generate the S boxes on the fly.

6. This algorithm, and the structure of the S boxes of Tiger, were chosen in a way which reduces linear and differential properties, and similarities of these properties in the four S boxes (between other things, by reducing the number of S boxes, and making them independent, unlike some original idea we had, which was intended to reduce the total memory size of the S boxes, and by choosing the large S boxes, which reduce linear and differential properties).

7. The randomizing parameter is easy to remember.

Items 1-4 as well as item 6 address the security properties of the S-boxes while items 5 and 7 deal with the functional requirements of the S-boxes. It is claimed by the designers of Tiger that the use of randomly generated Boolean mappings increase the security of the hash function. No quantitative arguments are presented to support this statement.

The randomly generated Boolean mapping used in Tiger is more difficult to manipulate than the Boolean mappings constructed from Boolean functions which are applied in parallel. This is due to the design criteria which have to be met by the randomly generated Boolean mappings. The design criteria applied in the generation of the Boolean mappings used by Tiger has the effect that it is impossible to construct a collision for a given Boolean mapping. Furthermore, a change in a single input bit is likely to cause a difference in more than one output bit.

Thus, it appears that the use of well chosen, randomly generated Boolean Mappings, avoid the problems and potential weaknesses observed in the bitwise Boolean mappings. However, the use of randomly generated Boolean mappings may result in an increase of resource requirements, such as storage capability and processor time. It is recommended that random mappings are utilised for optimal security.

11.3.2 Rotation

The rotation operator is used by all members of the MD4 family of hash functions. Rotations may be described in a number of ways. In [48] a rotation is described as a bit permutation. The rotation of x by n bits may also be viewed as a linear feedback shift register of length d with a feedback polynomial

$$f(x) = x^d + 1$$

which is clocked n times.

The rotation operation provides diffusion of bits throughout the hash function. MD4 and MD5 contain a single rotation in each step. HAVAL, SHA and SHA-1 contain two rotations applied to different chaining variables in each step. The use of rotations may complicate the solution of the sets of difference equations obtained. In particular in SHA and SHA-1 the use of two rotations and the permanent effect of the second rotation impose certain limits on an attacker, since an attacker has to choose values for chaining variables, which are invariant to the rotations (see Chapter 9). It is shown in [14] and [58] that it is possible to counter these rotations and find solutions to the sets of difference equations containing these rotations. A similar result is shown in Chapter 10 for HAVAL. However, the absence of rotations would have reduced the workload for finding inner collisions considerably.

Thus, rotations complicate the task of an attacker (depending on the use of the rotation) and therefore contribute to the security of the hash function. The rotation amounts should be chosen to obtain optimal diffusion in the compress function of the hash function.

None of the dedicated hash functions based on MD4 utilises data dependent rotations. It is unknown if data dependent rotations add or subtract to the security of dedicated hash functions. It may be possible for an attacker to exploit the data dependence. From a security point of view, it is recommended that caution be exercised if data dependent rotations are used.

11.3.3 Message Word Reuse

The iterative hash scheme employed in the MD4 family of hash functions requires that a message, M, is segmented into a number of message blocks, M_i. Each message block is

processed by the compress function $f()$. The compress functions of the MD4 family of hash functions divide the message block, M_i, into a number of message words. Each bit of each of these message words is re-used at least twice by each of the compress functions used by the MD4 family of hash functions. Two techniques of message re-use are employed by the MD4 family of hash functions. The first technique applies a permutation which changes the order in which the message words are accessed in consecutive rounds. The second technique applies a message expansion algorithm which derives new message words from the original message words. Both techniques are considered in this section.

Message Permutation

Each of the hash functions derived from MD4 takes a message block of fixed length as input. The message block is then divided into message words of 32 bits each. The message block is used in every round of the hash function. The order in which the message words are processed in each of these rounds are determined by the message permutation. MD4, MD5 and HAVAL employs message permutations. By exploiting the order in which message words are accessed in consecutive rounds, an attacker may obtain a set of difference equations. If this set of difference equations are solvable it may be possible to find collisions for the hash function. If message permutations are to be used care should be taken to choose the permutations in such a manner as to prevent solvable difference equations to be derived. This may prove difficult, if it is remembered that it is possible to obtain a solvable difference equation set representing 12 consecutive steps in MD5. Consequently it is advised that message permutations are avoided in the construction of dedicated hash functions.

Message Expansion

SHA and SHA-1 uses a message expansion algorithm instead of a fixed message permutation. The properties of the message expansion algorithm used by SHA is described in Chapter 9. From the analysis of SHA and SHA-1 it appears that the use of a message expansion algorithm adds to the effort required to derive a set of difference equations. The danger exist that an attacker may manipulate the message expansion algorithm to find one or more possible sets of solvable difference equations. An example of how the message expansion algorithm used by SHA may be manipulated is presented in Chapter 9.

It is noted that the dedicated hash functions based on the MD4 construction may be used as block ciphers. Specifically the initial value may be taken as the plaintext and the message block taken as the key. The hash result then represents the ciphertext. Thus, if we use SHA in this manner, we obtain a block cipher with a 160 bit block length (the initial value) and a 512 bit key (the message word). [1] The analogy may be extended to the message expansion algorithm. In block cipher terminology the message expansion algorithm is the equivalent of the key schedule. It is known that weak key schedules weaken the associated block cipher [64], [65]. In particular weak key scheduling algorithms allow related key attacks [65]. These attacks exploit a chosen difference between two unknown keys and allow the recovery of the keys given a number of known or chosen plaintexts. A weak message expansion algorithm may weaken the associated hash function by allowing the construction of two or more messages, which upon expansion, exhibit a specified difference pattern. This may allow the derivation of a set (or sets) of solvable difference Equations, which result in collisions. Thus both strong key scheduling algorithms and strong message expansion algorithms attempt to limit the extent to which an attacker may exploit specified differences between two distinct keys or messages. It is therefore suggested that the message expansion algorithm should meet the same requirements as those set for key scheduling algorithms.

In [66] a number of design criteria for key scheduling algorithms are proposed. In particular it is advised that the key schedule provides some guarantee of key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion. In addition each bit should be used by round $\frac{R}{2}$ of a R-round cipher. In [67] it is stated that the key schedule should have a high diffusion, and should behave irregularly with regard to the components of the round function of the block cipher.

In [67] a distinction is made between two kinds of key schedules namely pseudo-random key generation and key evolution. In the pseudo random approach the cipher key is used to seed a pseudo-random noise generator. The output of the pseudo-random noise generator serves as the round key. The relationship between the cipher key and the round keys generated in this way are complex. However, these schemes are slow and keys cannot be generated online (during encryption). Thus, pseudo-random round key generation schemes incurs a performance penalty. The key evolution strategy uses the cipher key as key to the first round and derives each round key from the previous round key by means of a transformation ψ.

[1] It should be noted that it is not recommended to use hash functions in this mode. Dedicated hash functions are not designed to be used as secret key encryption algorithms, and are likely to exhibit characteristics which make them susceptible to linear and differential cryptanalysis.

The process of key evolution is described by:

$$\begin{aligned} k^0 &= k \\ k^i &= \psi(k^{i-1}). \end{aligned}$$

The transform ψ may be described by bit permutations, rotations or elements from coding theory and abstract algebra. The advantage of this scheme is that it is fast and the round keys may be derived online. A disadvantage is that the underlying structure of the transform ψ may be exploited by an attacker.

From the preliminary study of SHA and SHA-1 it is observed that the key evolution strategy is used by the message expansion algorithms. As noted above it was shown that the underlying structure of the message expansion algorithm may be exploited to enable the construction of collisions for a limited number of rounds (see Chapter 9). In both [66] and [67] the importance of diffusion of key bits in the key scheduling algorithm is stressed. It is noted that the diffusion in the message expansion algorithm used by SHA is poor. Although the diffusion properties are improved by the addition of a rotation operator in SHA-1, it is unknown if it provides sufficient protection.

11.3.4 Addition mod 2^{32}

Addition mod 2^{32} is used by all the dedicated hash functions based on MD4. The addition mod 2^{32} contributes to the avalanche effect in the dedicated hash function. The contribution to the avalanche effect is ascribed to the propagation of the carry bit in addition operations.

11.3.5 Additive Constants

Additive constants are used by all members of the MD4 family of hash functions. MD4, SHA, SHA-1 and HAVAL use different constants in each round, while MD5 uses a different additive constant for each step. The use of an additive constant contributes little to the collision resistant property of the hash function. The attacks by Dobbertin, which results in collisions for hash functions require the derivation and solution of sets of difference equations. The constants are easily cancelled from these difference equations, and therefore does not contribute to the difficulty of solving the set of difference equations. The use of different

additive constants in each round does add to the pre-image resistance of a hash function. However the use of a different additive constant for each step requires additional storage capabilities and is considered unjustified if compared to the increase in security obtained (especially in terms of collision resistance).

11.3.6 Composition

A single step in each of the dedicated hash functions belonging to the MD4 family is composed by combining addition $\mod 2^{32}$, rotation operations and Boolean mappings or S-boxes. These steps are repeated a number of times (at least 48 times). A number of these steps represent a single round of the hash function. Each step compresses n 32-bit variables to a single 32 bit variable. In [57] the use of genetic algorithms to find solutions to expressions of the form encountered in dedicated hash functions is investigated. From the results in [57] it is observed that a single step in a dedicated hash function is neither collision resistant nor pre-image resistant. It is more difficult to find collisions for a number of these steps used iteratively. However it was shown that for MD4 [14] and MD5 [12] it is possible to find collisions for these iterated structures.

The Damgård-Merkle constructions [22] [23] shows that a collision resistant hash function may be constructed from a collision resistant function. The general design of the MD4 family resembles the iterative structure of the Damgård-Merkle construction. The security offered by the Damgård-Merkle construction would be attainable if the compress function of the dedicated hash functions are collision resistant. It is observed that the compress functions of these dedicated hash functions are themselves constructed by iteratively applying a number of similar steps. However, the ability to construct collisions for single steps and even a number of consecutive steps, (cryptanalysis of MD4 and MD5) implies that the level of security offered by the Damgård-Merkle construction can not be attained. It should however be noted that this does not imply that collisions are easily constructed for these hash functions.

11.4 CONCLUSION

In this chapter the basic building blocks encountered in dedicated hash functions are considered. Observations regarding the contribution of these building blocks to the security of dedicated hash functions derived from MD4 are made. The security offered by Boolean

mappings are considered. It is believed that randomly generated Boolean mappings offer more security than the bitwise Boolean mappings currently employed by the majority of the members of the MD4 family. The functional advantages of using bitwise Boolean mappings rather than randomly generated Boolean mappings outweighed the additional security obtained from randomly generated Boolean mappings in the design of the MD4 family (with the exception of Tiger). Rotations complicate the task of the cryptanalyst considerably and is considered a valuable building block. Specific attention is given to the message expansion algorithm. It is observed that the message expansion algorithm is similar to key schedule algorithms. Based on this observation and the results obtained from the analysis of the message expansion algorithms used by SHA and SHA-1, it is proposed that message expansion algorithms should exhibit properties similar to that required for key scheduling algorithms. Specific attention needs to be given to the diffusion properties of message expansion algorithms. Additive constants contribute little to the collision resistance property of the hash functions considered. Addition $\bmod 2^{32}$ adds to the diffusion process in the hash function due to the properties of the carry bit. The building blocks used in the MD4 family are combined into a single step, which is used iteratively. It is known that the individual steps are not collision resistant and consequently it is not known if the compress function constructed from these individual steps is collision resistant. If the step functions could be replaced by collision resistant one way functions, the resulting compress function would also be collision resistant and one-way according to the Damgård-Merkle construction. It is not known if collision resistant one way functions exist.

CHAPTER 12: CONCLUSION

12.1 DISCUSSION

Cryptographic hash functions are important primitive building blocks in information security. These functions form the corner stone of numerous authentication protocols, encryption algorithms and digital signatures. These cryptographic primitives are vital for creating a secure electronic commerce environment. Electronic commerce protocols such as SET and EMV rely on the existence of cryptographic hash functions. The two properties that make hash functions indispensable in cryptographic applications, are collision resistance and one-wayness. Throughout this dissertation we paid specific attention to the property of collision resistance. The property of collision resistance is vital for the non-repudiation service obtained through digital signatures.

However, designing hash functions that exhibit this property has proven to be extremely difficult. In the period from 1990 to 1994 a number of practical cryptographic hash functions were designed and implemented. These cryptographic hash functions include MD4, MD5, SHA, SHA-1, HAVAL, RIPEMD-128 and RIPEMD160. It was thought that these algorithms exhibited the properties of collision resistance and one-wayness. However in 1996 Dobbertin demonstrated that MD4 is not a collision resistant hash function. Within months the result was extended to RIPEMD-128 and MD5. One of the objectives of this dissertation was to generalise these attacks, apply it to other hash functions, and then derive design criteria that will defeat the generalised attack.

In pursuit of this objective a general introduction to cryptographic hash functions is presented in Chapters 1, 2, 3, 4. Chapter 1 introduced the relevant definitions and concepts surrounding cryptographic hash functions. Once the relevant concepts and definitions were in place, a number of generic attacks against cryptographic hash functions were considered in Chapter 3. Based on the definitions in Chapter 1 and the generic attacks presented in Chapter 3, a number of high level functional and security requirements were formulated. Given these requirements, a number cryptographic hash function designs were reviewed, including the MD4-family of functions.

12.2 RESULTS

After introducing the relevant concepts and definitions regarding cryptographic hash functions, a detailed description and re-construction of the attack on MD4 as formulated by Dobbertin is presented in Chapter 6. Using a novel approach, an alternative solution is presented which illustrates that a speed-up factor of 64 of the attack on MD4 as formulated by Dobbertin, can be achieved. In Chapter 7 the attack on MD5 as formulated by Dobbertin is considered. The attack is reconstructed from the source code used by Dobbertin to construct the collisions for MD5. Of particular interest are the techniques used to solve the non-linear Boolean equations. In Chapter 8 the attacks on MD4 and MD5 are generalised. The generalised attack presents a framework for the analyses of all iterated hash functions. In Chapters 9 and 10 the generalised attack is applied to SHA and HAVAL. It is shown that the generalised attack can be applied to reduced versions of SHA and HAVAL. The new results obtained for the HAVAL hash function indicates that three round HAVAL should not be used for cryptographic applications. To the best of our knowledge this is the first cryptanalytical result that has been published regarding the HAVAL cryptographic hash function.

In Chapter 11 we conclude this dissertation by presenting design criteria for dedicated cryptographic hash functions. The design criteria are based on the lessons learned from the analysis of MD4, MD5, SHA, SHA-1 and HAVAL. It is the intention that the application of these design criteria will defeat the generalised attack presented in Chapter 8.

12.3 SUMMARY AND FUTURE WORK

In this dissertation the attacks on MD4 and MD5 were generalised and applied to the SHA and HAVAL hash functions. Design criteria were proposed to defeat this generalised attack. It remains to determine the full extent to which the generalised attack presented in Chapter 8 can be applied to a number of dedicated cryptographic hash functions. It may prove interesting to apply this technique to a number of the Advanced Encryption Standard (AES) candidates to determine the difficulty of obtaining collisions for these cryptographic primitives. Another topic of interest lies in the design of message/key expansion algorithms. The use of strong diffusion structures such as those proposed by Massey in such a design may prove to be a challenging and interesting topic.