CHAPTER 5: GENERAL DEDICATED HASH FUNCTION CONSTRUCTIONS

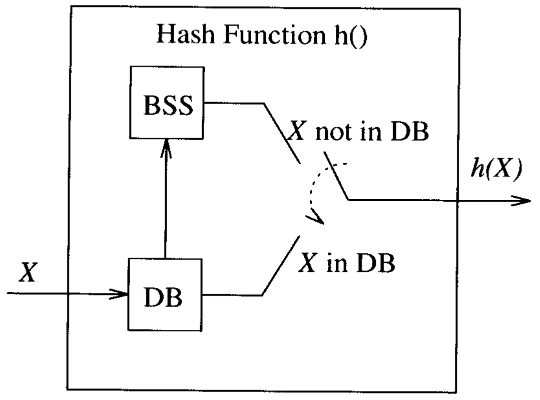
5.1 INTRODUCTION

This chapter introduces the notion of an ideal cryptographic hash function construction. This is an impractical construction and consequently the notion of the iterated cryptographic hash function is introduced. This construction was independently introduced by Dåmgard and Merkle and is commonly used in the construction of dedicated cryptographic hash functions. Dedicated hash function constructions based on this construction include the MD4 family of hash functions.

5.2 IDEAL CRYPTOGRAPHIC HASH FUNCTION

In [48] a construction is proposed for a super or ideal cryptographic hash function. The proposed construction is not dependent on a secret key, but can easily be extended by adding a key dependent element to the construction. In accordance to the definition of a hash function in Chapter 1, an input of variable length results in a hash value of a fixed length of n bits.

The construction consists of a database and a binary symmetric source. The message, X, is submitted to the hash function. The database is searched for the submitted message. If the message is found, the hash value associated with the message, $h(X)$, is presented as an output. If the message is not found in the database, the binary symmetric source generates a binary string of n bits. This binary string is presented as the hash value $h(X)$. The message, X, and the newly generated hash function, $h(X)$, is stored in the database for future use. Thus, the ideal cryptographic hash function is secure in the sense of cryptographic hash functions. A representation of this construction is shown in Figure 5.1.



DB: Database
BSS: Binary Simmetric Source

Figure 5.1: Ideal Cryptographic Hash Function

The proposed construction is impractical for the following reasons.

1. An infinite number of messages exists, and consequently infinite storage space is required for the database.
2. Due to the use of binary symmetric source all users should have access to the same database or hash function. This is impractical over large, distributed networks (such as the Internet).
3. A binary symmetric source is difficult to construct.

The construction for an ideal cryptographic hash function is reminiscent of the one time pad or Vernam cipher. Both the one time pad and the ideal cryptographic hash function are considered impractical. The goal of stream cipher design is to simulate certain properties of a one time pad while avoiding those properties which make it impractical. Likewise, when designing cryptographic hash functions, the goal should be to simulate certain properties of the ideal cryptographic hash function while circumventing the properties which are considered impractical.

The important design requirement of apparent independence between the message and the hash value was deduced from this construction (Chapter 4). Various building blocks that facilitate the construction of a hash function that satisfies this requirement, is presented in Section 5.5.

5.3 ITERATED HASH FUNCTIONS

The Damgård-Merkle scheme forms the basis of the majority of known hash functions. This scheme was independently proposed in [22] and [23]. It is an iterated scheme and hash functions constructed according to this scheme are referred to as iterated hash functions [49]. The following three components are identified in the Damgård-Merkle scheme:

1. The segmentation and padding rule.
2. The compress function.
3. The chaining rule.

5.3.1 The Segmentation and Padding Rule

By definition the hash function should hash a message of arbitrary length to a fixed length. The segmentation and padding feature of the Damgård-Merkle scheme allows the hashing of messages of arbitrary length, which is one of the functional requirements.

The segmentation rule is used to divide a message of an arbitrary length into blocks of fixed lengths. No special segmentation rules are known to exist. When segmentation is required the message is simply processed in a serial manner, dividing the message into blocks of a given length. The fixed block length is referred to as the elementary block length. If the message is not a multiple of the elementary block length, padding is required (see Figure 5.2).

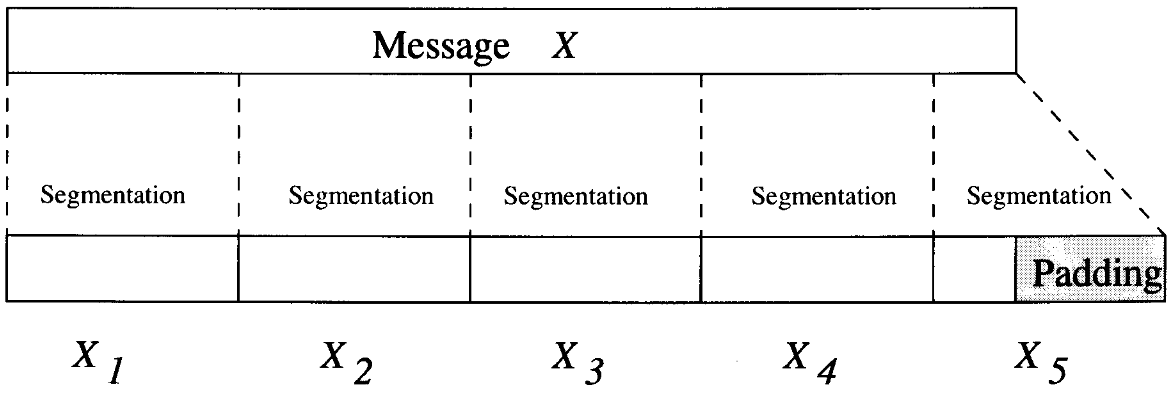


Figure 5.2: Segmentation and Padding

If the message is a multiple of the elementary hash length, padding is not required, but dependent on the applications, it is sometimes applied.

The padding rule is used to expand a message so that the message length is an exact multiple of the elementary block length on which the hash function operates. The padding rule can be used to add additional information to a message (redundancy). The redundancy provides additional security for the hash function against attacks (see Chapter 4). A number of padding rules have been proposed. A summary of these rules follows:

1. Pad the message with 0's until the padded message is a multiple of the block length. This padding rule is ambiguous, since it is not known how many of the trailing zeros are part of the message. This rule requires that either the length of the message be known, or that the message length is included in the message.
2. Pad the message with a single 1 followed, if necessary, by 0's until the padded message is a multiple of the required block length. If the message is a multiple of the required block length before padding commences, a block is added to the message. The additional block contains a single 1 followed by 0's.
3. Let z be a number of zeros and let r be the number of bits required for the binary representation of z. Pad the message with z zeros, except for the last r bits. Let the last r bits contain the binary representation of z. If less than r-bits remains in the last block, additional blocks are added until r can be appended to the zero padded message.
4. Let r be the number of bits required for the binary representation of the message length. Let b be the remaining number of bits in the last message block. Let z be the difference in the number of bits between b and r. Pad the message with z zeros, except for the last r bits. Let the last r bits contain the binary representation of the message length. If less than r-bits remains in the last block, additional blocks are added until r can be appended to the zero padded message.

The choice of padding rule depends on the application. However, padding rule number 4 offers the most security. This rule prevents an attacker from deleting or adding message blocks. If the attacker is an active eavesdropper, the threat posed by attacks such as the block correcting attack and fixed point attacks are minimised. It does however not necessarily prevent a legitimate participant from constructing messages of equal length using the fixed point and block correcting attacks.

In [23] Damgård presents a proof that if the round function, $f()$, is a collision resistant function (CRF), the construction described results in a collision resistant hash function. This proof holds only if the message length is appended to the message before hashing [3]. A variant of padding rule 4 is used for dedicated hash functions such as MD4, MD5, SHA and SHA-1.

For these reasons, padding rule number 4 or a variant thereof, is suggested for use in cryptographic hash functions.

5.3.2 The Compress Function

The second building block is the compress function or round function. The compress function, $f()$, reduces an input block X_i of m bits to a block of n bits. The compression function is the heart of the hash function since this is where the reduction of the message length occurs.

Damgård proved that generating a collision for a hash function based on the iterated scheme requires that either a collision has to be generated for the round function, $f()$, or a problem has to be solved of comparable difficulty. The proof is given in the framework of computational complexity theory. This effectively implies that the conditions and requirements imposed on cryptographic hash functions are transferred to the round function $f()$. In [3] it is stated that $f()$ should be a bijective function. This statement is based on the results obtained in [30].

The definitions for MACs, OWHFs and CRHFs can be modified and applied to the round functions used for these functions as follows:

Round Function for MAC

Definition 5.1 *The round function for a MAC is a function f() satisfying the following conditions:*

1. *The description of f() must be publicly known and the only secret information lies in the key, K, (extension of Kerkhoff's principle).*

2. *The key K should be used in every application of f() to every block of X_i.*

3. *The argument X_i is a segment of the message X. X_i has a fixed length of m bits and the result $f(K, X_i)$ has a fixed length of n bits.*

4. *Given f(), K and X_i, the computation of f(K,X) must be easy.*

5. *Given f() and X_i, it is hard to determine $f(K, X_i)$ with a probability of success significantly higher than 2^{-n}. Even where a large set of pairs $\{X_i, f(X_i, K)\}$ is known, where X_i have been selected by the opponent, it is "hard" to determine the key, K, or to compute $f(K, X_i')$ for any $X_i \neq X_i'$.*

Note the explicit requirement that the secret key should be used in each application of $f()$. This requirement is stated to discourage the use of the initial value, IV, as a secret key, K, in a MAC. If the IV is used as the key K, the key is used only in the first iteration of $f()$. This allows an attacker to add message blocks and update the hash value without knowledge of K. In certain hash functions, such as MD4, knowledge of the hash value and the message allows an attacker to determine K if K is used as the IV. For this reason it is advised that $f()$ is dependent on K.

Round Function for OWHF

Definition 5.2 *The round function for a OWHF is a function f() satisfying the following conditions:*

1. *The description of f() must be publicly known and should not require any secret information for its operation (extension of Kerkhoff's principle).*

2. *The argument X_i is a segment of the message X. X_i has a fixed length of m bits and the result $f(X_i)$ has a fixed length of n bits.*

3. *Given f() and X_i, the computation of $f(X_i)$ must be easy.*

4. *The hash function must be one way in the sense that:*

   (a) *given a Y in the image of f(), it is "hard" to find a message X_i such that $f(X_i) = Y$.*

(b) *given X_i and $f(X_i)$ it is "hard" to find a message block $X_i' \neq X_i$ such that $f(X_i) = f(X_i')$.*

Round Function for CRHF

Definition 5.3 *The round function for a OWHF is a function f() satisfying the following conditions:*

1. *The description of f() must be publicly known and should not require any secret information for its operation (extension of Kerkhoff's principle).*
2. *The argument X_i is a segment of the message X. X_i has a fixed length of m bits and the result $f(X_i)$ has a fixed length of n bits.*
3. *Given f() and X_i, the computation of $f(X_i)$ must be easy.*
4. *The hash function must be one way in the sense that:*
   (a) *given a Y in the image of f(), it is "hard" to find a message X_i such that $f(X_i) = Y$.*
   (b) *given X_i and $f(X_i)$ it is "hard" to find a message $X_i' \neq X_i$ such that $f(X_i) = f(X_i')$.*
5. *The round function f() must be collision resistant: This means that it is hard to find two distinct messages that result in the same image for the round function f().*

Thus, the conditions imposed on a round function used in a MAC, OWHF or CRHF, are similar to those imposed on the respective cryptographic hash functions. It is also interesting to note that the conditions imposed on the round functions are similar to those defined in [3] for one way functions (OWF) and collision resistant functions (CRF). A number of frequently used building blocks used in the construction of round functions for secure hash functions are identified in Sections 5.4 and 5.5.

A large number of attacks on hash functions based on the Damgård-Merkle Scheme, focuses on the compress function [17]. Although attacks on the compress function are usually specific to the hash algorithm, the general attacks described in Chapter 3 are also applicable. The parameters of the compress function should be chosen to render these attacks harmless.

5.3.3 The Chaining Rule

The third building block is the chaining rule. Chaining is used when the message length exceeds the maximum allowable input length to the compress function.

When processing the message in blocks, the previous result of the compress function has to be taken into account. This is accomplished by feeding the result from the previous compress operation back and combine it in some way with the new block that has to be processed.

The chaining rule determines which part of the chaining variable should be fed back. This approach is often used when a block cipher is used as a round function, since many block ciphers has a key length that is shorter than the block length. It is advised that the full result is fed back and used in the next iteration of the compress function.

Note that the introduction of chaining, allows attacks dependent on the chaining. These attacks include meet in the middle attacks, correcting block attacks, fixed point attacks and differential attacks (see Chapter 3 Section 3.4). The length of the chaining variables, measured in bits, should therefore be chosen to render these attacks computationally infeasible.

5.3.4 Construction

The interaction of the building blocks identified in the Damgård-Merkle scheme are described as follows. For a hash function $h()$ with a compress function $f()$ an initial value IV and a suitably padded message X, the interaction of the various building blocks are described by:

$$\begin{aligned} H_0 &= IV \\ H_i &= f(X_i, H_{i-1}) \qquad i \in \{1, 2, 3 \ldots j\} \\ h(X) &= H_j. \end{aligned}$$

A graphical representation of the interaction of the three building blocks are shown in Figure 5.3.

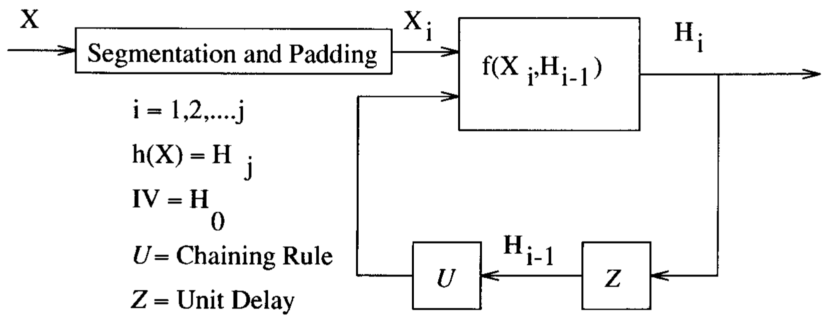


Figure 5.3: General Hash Function Construction (MDCs)

The construction shown in Figure 5.3 is specific to MDCs. For a MAC the interaction of the compress function, $f()$, the initial value, IV, the secret key, K, and a suitably padded message, X, is described as follows:

$$\begin{aligned} H_0 &= IV \\ H_i &= f(K, X_i, H_{i-1}) \quad i \in \{1, 2, 3 \ldots j\} \\ h(X) &= H_j. \end{aligned}$$

A graphical representation of the interaction of the various components of an iterated hash function used as a MAC, is shown in Figure 5.4.

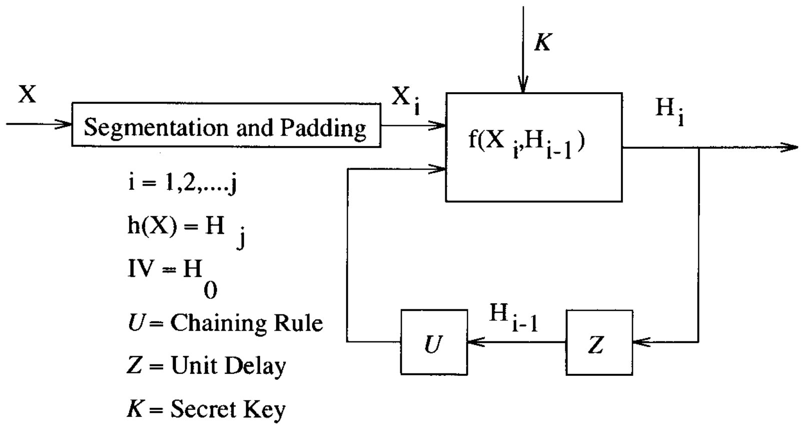


Figure 5.4: General Hash Function Construction (MACs)

A number of the better known dedicated hash functions based on the Damgård-Merkle scheme include BCA, MD4, MD5, SHA, SHA-1, Haval, RIPEMD, N-Hash, Snefru and Tiger.

5.4 ROUND FUNCTION CONSTRUCTIONS

The majority of hash functions designed in recent years make use of an iterative structure (Section 5.3). Hash functions based on iterative structures require secure round functions. A number of conditions are imposed on the round functions (Section 5.3). Currently there are three round function constructions in popular use. They are:

1. MD4-Family Construction.
2. Block Ciphers.
3. Stream Ciphers.

In this chapter the MD4-family construction is considered in detail. The use of block and stream ciphers in round function constructions are considered in Appendix A.

5.4.1 MD4-Family Construction

The round 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-1 [13], Tiger [47] and RIPEMD-160 [15]. The round functions of these dedicated hash functions are similar in design and construction.

Consider the iterated hash function as represented in Figures 5.3 and 5.4. Note that at most three inputs are supplied to the round function. These inputs consist of the current message block, X_i, the previous hash result, H_{i-1}, and a secret key, K. Note that the secret key is only applicable when the construction is used as a MAC. The generalised MD4-family construction does not allow for the inclusion of a secret key. Adaptations of this construction that does make allowance for a secret key is presented in Appendix A.

The round function used in the MD4-family of constructions is itself an iterated construction. The round function take as input the previous hash result H_{i-1} and the current message block, X_i (see Figure 5.5).

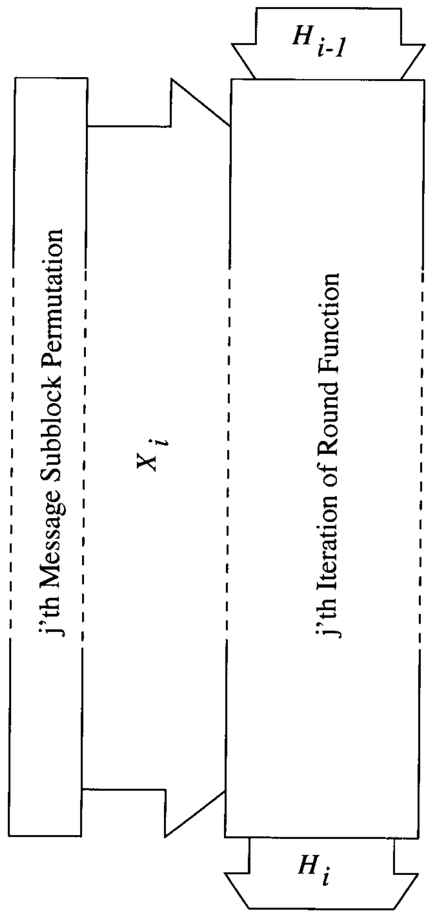

H_{i-1}
j'th Message Subblock Permutation
X_i
j'th Iteration of Round Function
H_i

Figure 5.5: Generalised MD4 Round Function Construction

The message block X_i is segmented into k sub-blocks. The previous hash result is set equal to the initial chaining variable, C, for the round function. The set of message sub-blocks are permutated and applied to the j'th iteration of the round functions. The chaining variable, C, is then updated and applied to the next iteration of the round function. This process is repeated three or four times. The permutation of the sub-block and the method used for updating the chaining variable for each round is different for each iteration in the round function. Each iteration of the round function is constructed from the elementary building blocks described in Section 5.5.

5.5 ROUND FUNCTION BUILDING BLOCKS

This section contains descriptions of the building blocks frequently used in the construction of round functions for cryptographic hash functions. These building blocks facilitate the fulfillment of the requirements of diffusion and confusion as defined by Shannon [46] (see Chapter 4 Section 4.3.1). For this reason the building blocks identified in this section are also commonly used in the construction of other cryptographic functions.

5.5.1 Bit Permutations

The use of bit permutations as building blocks in cryptographic primitives are considered in [48]. A bit permutation of a vector modifies the order of the components of the vector, without changing the values of the individual bits. In [48] it is recommended that the new bit positions should be calculated from the old bit positions using a simple expression. As an example, consider the simple cyclic rotation over a vector of length l. The bit in position i is rotated over d-positions and the new bit position is given by $i + d \bmod l$. These permutations are popular and are used in a number of dedicated hash functions, including MD4, MD5, SHA and SHA-1.

If the bit permutation is implemented in dedicated hardware, the bit permutations can be achieved through "hardwiring" the permutation into silicon. The price paid for this approach is the large amount silicon required to accomplish such a task [48].

If the bit permutation is to be implemented in software on general purpose processors, a permutation where individual bits are to be moved around should be avoided due to the reduction in performance. Lookup tables can be used to speed-up the bit permutation process, but as remarked in [48] the size of the lookup tables grows exponentially, rendering this technique infeasible. Instead it is advised that bit permutations be implemented in a blockwise manner. Bit permutations which satisfy this requirement include vector rotations, as described earlier. Note that the choice of a specific vector length, say 32 bits, is likely to favour certain architectures, while putting others at a disadvantage. Thus, portability of the algorithm is decreased.

As mentioned previously, vector rotations form popular building blocks for a number of well known dedicated hash functions. The degree of security obtained from the use of these vector rotations is considered inadequate in certain dedicated hash functions. The use of vector rotations in MD4 in particular is shown to add little in terms of additional security [14], [17]. One of the reasons the vector rotations in MD4 are ineffective is that the rotation factor, d, is a constant for each step. With the rotation constants known, the effect of the rotations can be calculated and countered. In the RC5 and RC6 encryption algorithm data dependent vector rotations are introduced [27], [50]. It is therefore conceivable that security obtained from vector rotations can be increased by making the rotation factor, d, data dependent when designing dedicated hash functions.

Thus, when choosing a bit permutation the method of implementation, the portability of the algorithm, the reduction in speed for a complicated bit permutation technique and the required cryptographic strength should be kept in mind.

5.5.2 Bitwise Boolean Operations

Bitwise Boolean operations are widely used in the MD-family of hash functions (MD4, MD5 and SHA-1). A bitwise Boolean operation treats all individual components of the binary vector in the same manner. Commonly used bitwise operators are complementation, bitwise AND, OR and XOR.

Bitwise binary operators are easily described in both hardware and software. For a processor architecture with word length k, the bitwise Boolean operation on a vector of length n can be split into $\lceil \frac{n}{k} \rceil$ operations, if $n > k$. For these reasons bitwise Boolean operations are portable, not only between different processor architectures, but also between hardware and software platforms.

A number of desirable properties of Boolean functions are proposed in chapter 3 of [51]. These properties should be used as design criteria when constructing bitwise Boolean operations for cryptographic hash functions.

5.5.3 Substitution Boxes

A substitution box, or S-box, is defined as an $n \times m$ mapping of a n bit vector to an m bit vector, m and n need not be equal [51].

S-boxes are traditionally used as building blocks in block ciphers and stream ciphers. S-boxes have not been used extensively in the design of cryptographic hash functions. An example of a dedicated hash function that makes use of S-boxes is Tiger [47].

For ease of implementation and description it is proposed to keep S-boxes small, specify a way to generate the S-boxes at run time, or limit the number of S-boxes used [48]. When using hardware implementations the S-boxes should be kept small due to the silicon area required. In software S-boxes are usually contained in arrays. This limits portability due to specific data word lengths used by specific processors.

In chapter 4 of [51] it is remarked that if $m = 1$ the mapping is a Boolean function. Thus a Boolean function is a special instance of a S-box. The MD family of hash functions makes use of bitwise Boolean functions rather than S-boxes. This choice is due to the memory and performance penalties associated with the use of large S-boxes. It is believed that the use of cryptographically strong S-boxes instead of bitwise Boolean operators will result in stronger cryptographic hash functions [47]. An extensive treatment of the issue of S-box design and analysis is given in [51].

5.5.4 Modular Arithmetic Operations

Modular arithmetic has been identified as a building block from which hash functions can be constructed [48]. A number of hash functions have been based on modular arithmetic [2]. In [2] three arguments are presented in favour of using modular arithmetic:

1. Hardness of number theoretic problems.
2. Availability of modular arithmetic implementations.
3. Scalability.

The schemes based on modular arithmetic are classified according to the size of the modulus used. Schemes with a small modulus (32 bits) have been proposed in [52]. These schemes are believed vulnerable to divide and conquer attacks [3]. Schemes with a large modulus (512 bit or more) are evaluated in [3]. It has been shown in [40] that these schemes are insecure when used with the RSA signature scheme. The use of modular arithmetic for the construction of cryptographically strong hash functions is considered limited [2].

5.6 CONCLUSION

In this chapter, a generic construction for building MACs and MDCs that satisfies the requirements presented in Chapter 4, was introduced. In particular the construction of the iterated hash scheme used by the MD4 family of functions were considered. The design of appropriate round functions is considered in Section 5.4 and 5.5. Commonly used building blocks for cryptographic primitives are discussed in Section 5.5.

CHAPTER 6: ANALYSIS OF THE MD4 HASH ALGORITHM

6.1 INTRODUCTION

In this chapter the MD4 algorithm is considered. The MD4 algorithm is described followed by the reconstruction of the analysis of MD4 as presented by Dobbertin [14]. In addition the attack presented by Dobbertin is extended in a novel way that allows the computational requirements to be reduced by a factor 64.

6.2 INTRODUCTION TO MD4

MD4 is a dedicated hash function proposed by R. Rivest [10], [44]. MD4 is an acronym for "Message Digest 4". MD4 is an unkeyed dedicated cryptographic hash function (MDC). MD4 is based on the iterative construction proposed discussed in Chapter 5. The MD4 algorithm was designed to meet the following criteria.

1. Security.
2. Speed.
3. Simplicity and compactness.
4. Favors little endian architectures.

The most prominent design criterion is security. This implies that it should be computationally infeasible to find two messages, M_1 and M_2, that hashes to the same value. In other words, MD4 is intended to be a collision resistant hash function. The remaining three criteria are concerned with high speed implementation in software.

A complete definition of MD4, including the padding rule is given in [10] and [44]. Since this chapter is specifically concerned with the cryptanalysis of MD4 it is useful to consider the operation of MD4 before concentrating on the analysis.

6.3 NOTATION

Before proceeding to describe the operation of MD4 it is appropriate to define the notation used in this chapter.

$$\begin{aligned} X[j] &= \text{32-bit word,} \qquad j \in \{0,1,2\ldots15\} \\ \tilde{X}[j] &= \text{Alternative 32-bit word,} \quad j \in \{0,1,2,\ldots15\} \\ (AA,BB,CC,DD) &= \text{Hash variables} \\ (A_i,B_i,C_i,D_i) &= \text{Chaining variables after step } i, \quad i \in \{0,1,2,\ldots47\} \end{aligned}$$

6.4 THE MD4 ALGORITHM

MD4 is an iterated hash function. Each iteration requires the application of the compress function. For MD4 the compress function is defined by the sequential application of three distinct rounds. The elementary size of a message block is 512 bits. If the message is not a multiple of 512 bits, a padding rule is used. Before the message block is processed it is divided into 16 blocks of 32 bits each. Four 32-bit chaining variables are used to produce a 128-bit hash value. The following steps are identified in the MD4 algorithm.

Algorithm 6.1 *MD4 hash algorithm*

1. *Pad message.*
2. *Append the length of the message.*
3. *Hash and chaining variable initialisation.*
4. *Round 1.*
5. *Round 2.*
6. *Round 3.*
7. *Update hash variables.*
8. *Has the entire message been processed ?*
   (a) *No: Repeat from step 4.*
   (b) *Yes: Continue.*
9. *Output hash value.*

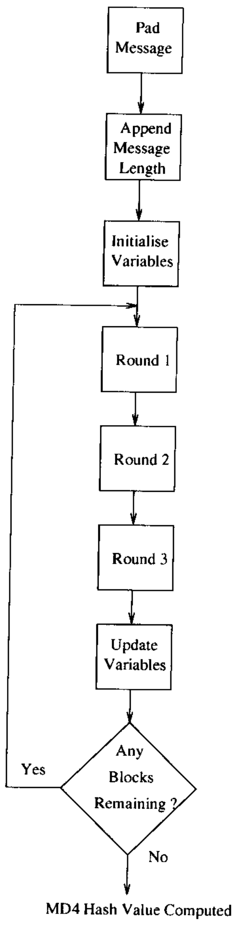


Figure 6.1: MD4 Block Diagram

A block diagram of the steps in the MD4 algorithm is shown in Figure 6.1. A description of each of the steps identified in the MD4 algorithm is presented next.

6.4.1 Message Padding

The first two steps ensure that the message length is a multiple of 512 bits. This allows the message to be processed in blocks of 512 bits at a time. The padding rule is described in [10] and [44].

6.4.2 Initial Values

Step 3 initiates the four chaining variables as follows:

$$\begin{aligned} A_0 &= \texttt{0x67452301} \\ B_0 &= \texttt{0xEFCDAB89} \\ C_0 &= \texttt{0x98BADCFE} \\ D_0 &= \texttt{0x10325476}. \end{aligned}$$

The hash variables contains the hash value for each iteration and is initialised as shown below:

$$\begin{aligned} AA &= A_0 \\ BB &= B_0 \\ CC &= C_0 \\ DD &= D_0. \end{aligned}$$

6.4.3 Iterative Rounds

Steps 4-8 performs the iterative computation of the hash value. The hash value is computed by applying three distinct rounds to each 512 bit block of the message. The hash function derives its strength from these three rounds. The hash variables are updated once all three rounds have been completed. If all of the 512 bit blocks have been processed the updated hash variables contains the final hash value.

The elementary operation within each round is described by:

$$\begin{aligned} R(S,T,U,V,X,K,W,r,j) &= (S + f_r(T,U,V) + X[j] + K_r)^{\lll W_{r_j}} \\ S &= R(S,T,U,V,X,K,W,r,j) \end{aligned}$$

with:

$$\begin{aligned}
S, T, U, V &= \text{32 bit words} \\
X[j] &= j\text{'th 32 bit word of the message,} \quad j \in \{0, 1, 2 \ldots 15\} \\
r &= \text{Round } r \quad r \in \{1, 2, 3\} \\
K_r &= \text{The constant for the r'th round} \\
f_r &= \text{The boolean function in the r'th round} \\
x^{\lll W_{r_j}} &= \text{Circular rotate } x \text{ left by} W_{r_j} \text{ bits} \\
x + y &= \text{Modulo } 2^{32} \text{ addition of } x \text{ and } y.
\end{aligned}$$

Each round differs from the other with regard to K_r and f_r. The index j in $X[j]$ is used to permutate the 512 bit input in 32-bit blocks for each round of the hash function. In each round W takes on one of four values.

Three boolean functions are defined for MD4.

$$\begin{aligned}
F(X, Y, Z) &= (X \wedge Y) \vee (\neg X \wedge Z) \\
G(X, Y, Z) &= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \\
H(X, Y, Z) &= X \oplus Y \oplus Z
\end{aligned}$$

with:

$\wedge$	=	Bitwise AND	$\neg$	=	Bitwise NOT
$\vee$	=	Bitwise OR	$\oplus$	=	Bitwise XOR.

The function $G(X, Y, Z)$ is a majority function. Thus for each bit position in X, Y, and Z the binary value that occurs more than once is selected. The function $F(X, Y, Z)$ is essentially a selection function. A graphical representation of the selection function $F(X, Y, Z)$ is shown in Figure 6.2.

The constants for K_r are defined in [10] as:

$$\begin{aligned}
K_1 &= \texttt{0x00000000} \\
K_2 &= \texttt{0x5A827999} \\
K_3 &= \texttt{0x6ED9EBA1}
\end{aligned}$$

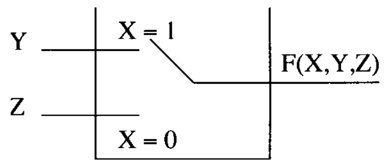


Figure 6.2: Selection Function $F(X, Y, Z)$

Three distinct rounds are identified for MD4. These rounds constitutes the compress function for MD4. It is therefore considered appropriate to give a detailed description of the rounds of MD4, since all the known attacks on MD4 focuses on the compress function. The equations describing the round functions are presented with this fact in mind.

Round 1

For the first round K_1 = 0x00000000 and is omitted from the equations. The boolean function $f_1 = F(X, Y, Z)$ for the first round. The four possible rotation constants for the first round of MD4 are defined as:

$$fs1 = 3 \qquad fs3 = 11$$
$$fs2 = 7 \qquad fs4 = 19$$

The complete set of equations for the first round are shown below:

$$\begin{aligned}
A_3 &= (A_0 + F(B_0, C_0, D_0) + X[0])^{\lll fs1} &(6.1)\\
D_3 &= (D_0 + F(A_3, B_0, C_0) + X[1])^{\lll fs2} &(6.2)\\
C_3 &= (C_0 + F(D_3, A_3, B_0) + X[2])^{\lll fs3} &(6.3)\\
B_3 &= (B_0 + F(C_3, D_3, A_3) + X[3])^{\lll fs4} &(6.4)\\
A_7 &= (A_3 + F(B_3, C_3, D_3) + X[4])^{\lll fs1} &(6.5)\\
D_7 &= (D_3 + F(A_7, B_3, C_3) + X[5])^{\lll fs2} &(6.6)\\
C_7 &= (C_3 + F(D_7, A_7, B_3) + X[6])^{\lll fs3} &(6.7)\\
B_7 &= (B_3 + F(C_7, D_7, A_7) + X[7])^{\lll fs4} &(6.8)\\
A_{11} &= (A_7 + F(B_7, C_7, D_7) + X[8])^{\lll fs1} &(6.9)\\
D_{11} &= (D_7 + F(A_{11}, B_7, C_7) + X[9])^{\lll fs2} &(6.10)\\
C_{11} &= (C_7 + F(D_{11}, A_{11}, B_7) + X[10])^{\lll fs3} &(6.11)\\
B_{11} &= (B_7 + F(C_{11}, D_{11}, A_{11}) + X[11])^{\lll fs4} &(6.12)\\
A_{15} &= (A_{11} + F(B_{11}, C_{11}, D_{11}) + X[12])^{\lll fs1} &(6.13)\\
D_{15} &= (D_{11} + F(A_{15}, B_{11}, C_{11}) + X[13])^{\lll fs2} &(6.14)\\
C_{15} &= (C_{11} + F(D_{15}, A_{15}, B_{11}) + X[14])^{\lll fs3} &(6.15)\\
B_{15} &= (B_{11} + F(C_{15}, D_{15}, A_{15}) + X[15])^{\lll fs4} &(6.16)
\end{aligned}$$

Round 2

For the second round K_2 takes on the value as defined previously. The boolean function $f_2 = G(X, Y, Z)$ for the second round. The four possible rotation constants for the first round of MD4 are defined as:

$$\begin{aligned} gs1 &= 3 & \qquad gs3 &= 9\\ gs2 &= 5 & \qquad gs4 &= 13 \end{aligned}$$

The complete set of equations describing the second round are shown below:

$$A_{19} = (A_{15} + g(B_{15}, C_{15}, D_{15}) + X[0] + K_2)^{\lll gs1} \tag{6.17}$$
$$D_{19} = (D_{15} + g(A_{19}, B_{15}, C_{15}) + X[4] + K_2)^{\lll gs2} \tag{6.18}$$
$$C_{19} = (C_{15} + g(D_{19}, A_{19}, B_{15}) + X[8] + K_2)^{\lll gs3} \tag{6.19}$$
$$B_{19} = (B_{15} + g(C_{19}, D_{19}, A_{19}) + X[12] + K_2)^{\lll gs4} \tag{6.20}$$
$$A_{23} = (A_{19} + g(B_{19}, C_{19}, D_{19}) + X[1] + K_2)^{\lll gs1} \tag{6.21}$$
$$D_{23} = (D_{19} + g(A_{23}, B_{19}, C_{19}) + X[5] + K_2)^{\lll gs2} \tag{6.22}$$
$$C_{23} = (C_{19} + g(D_{23}, A_{23}, B_{19}) + X[9] + K_2)^{\lll gs3} \tag{6.23}$$
$$B_{23} = (B_{19} + g(C_{23}, D_{23}, A_{23}) + X[13] + K_2)^{\lll gs4} \tag{6.24}$$
$$A_{27} = (A_{23} + g(B_{23}, C_{23}, D_{23}) + X[2] + K_2)^{\lll gs1} \tag{6.25}$$
$$D_{27} = (D_{23} + g(A_{27}, B_{23}, C_{23}) + X[6] + K_2)^{\lll gs2} \tag{6.26}$$
$$C_{27} = (C_{23} + g(D_{27}, A_{27}, B_{23}) + X[10] + K_2)^{\lll gs3} \tag{6.27}$$
$$B_{27} = (B_{23} + g(C_{27}, D_{27}, A_{27}) + X[14] + K_2)^{\lll gs4} \tag{6.28}$$
$$A_{31} = (A_{27} + g(B_{27}, C_{27}, D_{27}) + X[3] + K_2)^{\lll gs1} \tag{6.29}$$
$$D_{31} = (D_{27} + g(A_{31}, B_{27}, C_{27}) + X[7] + K_2)^{\lll gs2} \tag{6.30}$$
$$C_{31} = (C_{27} + g(D_{31}, A_{31}, B_{27}) + X[11] + K_2)^{\lll gs3} \tag{6.31}$$
$$B_{31} = (B_{27} + g(C_{31}, D_{31}, A_{31}) + X[15] + K_2)^{\lll gs4} \tag{6.32}$$

Round 3

For the third round K_3 takes on the value as previously defined. The boolean function $f_3 = H(X, Y, Z)$ for the third round. The four possible rotation constants for the first round of MD4 are defined as:

$$hs1 = 3 \qquad hs3 = 11$$
$$hs2 = 9 \qquad hs4 = 15$$

The complete set of equations describing the third round are shown below:

$$\begin{aligned}
A_{35} &= (A_{31} + h(B_{31}, C_{31}, D_{31}) + X[0] + K_3)^{\lll hs1} && (6.33)\\
D_{35} &= (D_{31} + h(A_{35}, B_{31}, C_{31}) + X[8] + K_3)^{\lll hs2} && (6.34)\\
C_{35} &= (C_{31} + h(D_{35}, A_{35}, B_{31}) + X[4] + K_3)^{\lll hs3} && (6.35)\\
B_{35} &= (B_{31} + h(C_{35}, D_{35}, A_{35}) + X[12] + K_3)^{\lll hs4} && (6.36)\\
A_{39} &= (A_{35} + h(B_{35}, C_{35}, D_{35}) + X[2] + K_3)^{\lll hs1} && (6.37)\\
D_{39} &= (D_{35} + h(A_{39}, B_{35}, C_{35}) + X[10] + K_3)^{\lll hs2} && (6.38)\\
C_{39} &= (C_{35} + h(D_{39}, A_{39}, B_{35}) + X[6] + K_3)^{\lll hs3} && (6.39)\\
B_{39} &= (B_{35} + h(C_{39}, D_{39}, A_{39}) + X[14] + K_3)^{\lll hs4} && (6.40)\\
A_{43} &= (A_{39} + h(B_{39}, C_{39}, D_{39}) + X[1] + K_3)^{\lll hs1} && (6.41)\\
D_{43} &= (D_{39} + h(A_{43}, B_{39}, C_{39}) + X[9] + K_3)^{\lll hs2} && (6.42)\\
C_{43} &= (C_{39} + h(D_{43}, A_{43}, B_{39}) + X[5] + K_3)^{\lll hs3} && (6.43)\\
B_{43} &= (B_{39} + h(C_{43}, D_{43}, A_{43}) + X[13] + K_3)^{\lll hs4} && (6.44)\\
A_{47} &= (A_{43} + h(B_{43}, C_{43}, D_{43}) + X[3] + K_3)^{\lll hs1} && (6.45)\\
D_{47} &= (D_{43} + h(A_{47}, B_{43}, C_{43}) + X[11] + K_3)^{\lll hs2} && (6.46)\\
C_{47} &= (C_{43} + h(D_{47}, A_{47}, B_{43}) + X[7] + K_3)^{\lll hs3} && (6.47)\\
B_{47} &= (B_{43} + h(C_{47}, D_{47}, A_{47}) + X[15] + K_3)^{\lll hs4} && (6.48)
\end{aligned}$$

A graphical representation of the three rounds that constitutes the compress function of MD4 is shown in Figure 6.3 (derived from [11]).

Update Variables

After completion of all three rounds, the hash variables are updated as shown below:

$$\begin{aligned}
AA = A_0 &= A_{47} + AA\\
BB = B_0 &= B_{47} + BB\\
CC = C_0 &= C_{47} + CC\\
DD = D_0 &= D_{47} + DD
\end{aligned}$$

If all of the 512 bit message blocks have been processed, the final hash value is given by AA,

BB, CC, and DD. If there are unprocessed message blocks remaining, A_0, B_0, C_0, and D_0 contains the new initial values for the next iteration of MD4.

6.5 CRYPTANALYSIS OF MD4

The MD4 hash function has been extensively analysed since its introduction in 1990 [10],[44]. In 1991 an attack on the last two rounds of MD4 was presented by Bosselaers and den Boer [11]. An unpublished attack on the first two rounds of MD4 is credited to Merkle. In 1994 Vaudenay published an attack on the first two rounds of MD4 [53]. In 1995 Dobbertin p-resented a technique to cryptanalyse the MD4 hash function [14]. The result included in this chapter builds on the results obtained by Dobbertin by presenting an algorithm which requires $O(2^6)$ times less iterations for finding a collision.

This contains selected results obtained from the cryptanalysis of MD4. The remainder of this chapter is organised as follows. First the notation used for the cryptanalysis of MD4 is introduced. A review of the attack in [14] is then presented. This is followed by a description of an alternative algorithm which can be used to speed up the attack proposed in [14]. The results obtained from the use of the alternative algorithm are then considered. The source code that implements this attack is included for reference purposes as Appendix C.

6.6 NOTATION

Before proceeding with a description of the cryptanalysis of MD4 the following notation is introduced.

$$\begin{aligned}
X[j] &= \text{32-bit word} \\
& \quad j \in [1,16] \\
M_1 &= \text{Message 1} \\
M_2 &= \text{Message 2} \\
Z_i &= \text{Chaining variable for } M_1 \text{ after step } i \\
\tilde{Z}_i &= \text{Chaining variable for } M_2 \text{ after step } i \\
& \quad Z \in \{A, B, C, D\} \\
& \quad i \in [0,47]
\end{aligned}$$

The relationship between M_1 and M_2 is given by:

$$\tilde{X}[12] = X[12] + 1 \tag{6.49}$$

In [14] the following notation is introduced.

$$\begin{aligned}
\text{COMPRESS}_z^y &= \text{Value of chaining variables after steps} \\
&\quad \text{y to z of the compress function is performed.} \\
\Delta_i &= (A_i - \tilde{A}_i, B_i - \tilde{B}_i, C_i - \tilde{C}_i, D_i - \tilde{D}_i) \\
&\quad i \in [0, 47] \\
Z^{\lll X} &= \text{Left circular rotation of Z by X bit positions.} \\
-Z^{\lll X} &= -(Z^{\lll X}).
\end{aligned}$$

The following relationship exists between the notation used in [14] and the notation used in this appendix.

$$\begin{array}{lcl@{\qquad}lcl@{\qquad}lcl}
 & & & V &=& D_{15} & A_* &=& A_{19} \\
B &=& B_{11} & \tilde{V} &=& \tilde{D}_{15} & B_* &=& B_{19} \\
C &=& C_{11} & W &=& C_{15} & \tilde{B}_* &=& \tilde{B}_{19} \\
U &=& A_{15} & \tilde{W} &=& \tilde{C}_{15} & C_* &=& C_{19} \\
\tilde{U} &=& \tilde{A}_{15} & Z &=& B_{15} & \tilde{C}_* &=& \tilde{C}_{19} \\
 & & & \tilde{Z} &=& \tilde{B}_{15} & D_* &=& D_{19}.
\end{array}$$

6.7 DOBBERTIN'S ATTACK: A REVIEW

The cryptanalysis of MD4 is described in [14]. The attack could be viewed as a divide and conquer attack. The attack is divided into two parts. The first part is concerned with the establishment of a so-called inner almost-collision. The second part of the attack is based on a differential attack and the matching of initial values. The differential attack can only succeed if the criteria set for the establishment of the inner almost-collisions has been met.

An inner almost-collision is obtained if a set of chaining variables are found for which the difference between COMPRESS$_{19}^{12}$ performed on M_1 and M_2 is:

$$\Delta_{19} = (0, 1^{\lll 25}, -1^{\lll 5}, 0)$$

In order for the differential attack to be successful the above condition has to be met. Thus obtaining an inner almost-collision is central to the success of the attack described in [14]. The above condition implies that the following relationship should hold between the chaining variables obtained for message M_1 and M_2 after step 19.

$$\begin{aligned} \tilde{A}_{19} &= A_{19} & \qquad \tilde{C}_{19} &= C_{19} + 1^{\lll 5} \\ \tilde{B}_{19} &= B_{19} - 1^{\lll 25} & \qquad \tilde{D}_{19} &= D_{19}. \end{aligned}$$

Using these relationships and conditions the following set of non-linear equations were derived in [14].

$$\begin{aligned}
1 &= \tilde{A}_{15}^{\lll 29} - A_{15}^{\lll 29} && (6.50)\\
F(\tilde{A}_{15}, B_{11}, C_{11}) - F(A_{15}, B_{11}, C_{11}) &= \tilde{D}_{15}^{\lll 25} - D_{15}^{\lll 25} && (6.51)\\
F(\tilde{D}_{15}, \tilde{A}_{15}, B_{11}) - F(D_{15}, A_{15}, B_{11}) &= \tilde{C}_{15}^{\lll 21} - C_{15}^{\lll 21} && (6.52)\\
F(\tilde{C}_{15}, \tilde{D}_{15}, \tilde{A}_{15}) - F(C_{15}, D_{15}, A_{15}) &= \tilde{B}_{15}^{\lll 13} - B_{15}^{\lll 13} && (6.53)\\
G(\tilde{B}_{15}, \tilde{C}_{15}, \tilde{D}_{15}) - G(B_{15}, C_{15}, D_{15}) &= A_{15} - \tilde{A}_{15} && (6.54)\\
G(\tilde{A}_{19}, \tilde{B}_{15}, \tilde{C}_{15}) - G(A_{19}, B_{15}, C_{15}) &= D_{15} - \tilde{D}_{15} && (6.55)\\
G(\tilde{D}_{19}, \tilde{A}_{19}, \tilde{B}_{15}) - G(D_{19}, A_{19}, B_{15}) &= C_{15} - \tilde{C}_{15} + \\
&\quad \tilde{C}_{19}^{\lll 23} - C_{19}^{\lll 23} && (6.56)\\
G(\tilde{C}_{19}, \tilde{D}_{19}, \tilde{A}_{19}) - G(C_{19}, D_{19}, A_{19}) &= B_{15} - \tilde{B}_{15} - 1 + \\
&\quad \tilde{B}_{19}^{\lll 19} - B_{19}^{\lll 19} && (6.57)
\end{aligned}$$

Once a solution to equations (6.50) to (6.57) have been obtained, the messages M_1 and M_2

that leads to the inner almost-collision can be found by solving equations (6.58) to (6.66):

$$\begin{aligned}
X[13] &= \text{Arbitrary} && (6.58)\\
X[14] &= C_{15}^{\lll 21} - C_{11} - F(D_{15}, A_{15}, B_{11}) && (6.59)\\
X[15] &= B_{15}^{\lll 13} - B_{11} - F(C_{15}, D_{15}, A_{15}) && (6.60)\\
X[0] &= A_{19}^{\lll 29} - A_{15} - F(B_{15}, C_{15}, D_{15}) - K_1 && (6.61)\\
X[4] &= D_{19}^{\lll 27} - D_{15} - F(A_{19}, B_{15}, C_{15}) - K_1 && (6.62)\\
X[8] &= C_{19}^{\lll 23} - C_{15} - F(D_{19}, A_{19}, B_{15}) - K_1 && (6.63)\\
X[12] &= B_{19}^{\lll 19} - B_{15} - F(C_{19}, D_{19}, A_{19}) - K_1 && (6.64)\\
D_{11} &= D_{15}^{\lll 25} - F(A_{15}, B_{11}, C_{11}) - X[13] && (6.65)\\
A_{11} &= A_{15}^{\lll 29} - F(B_{11}, C_{11}, D_{11}) - X[12]. && (6.66)
\end{aligned}$$

By setting:

$$A_{15} = \texttt{0xFFFFFFFF} \qquad \tilde{A}_{15} = \texttt{0x00000000} \qquad \tilde{B}_{11} = \texttt{0x00000000}.$$

Equations (6.50) to (6.57) are reduced to:

$$\begin{aligned}
\tilde{B}_{15} &= B_{15} - G(\tilde{C}_{19}, \tilde{D}_{19}, \tilde{A}_{19}) + G(C_{19}, D_{19}, A_{19}) + \\
&\quad \tilde{B}_{19}^{\lll 19} - B_{19}^{\lll 19} - 1 && (6.67)\\
\tilde{C}_{15} &= C_{15} - G(\tilde{D}_{19}, \tilde{A}_{19}, \tilde{B}_{15}) + G(D_{19}, A_{19}, B_{15}) + \\
&\quad \tilde{C}_{19}^{\lll 23} - C_{19}^{\lll 23} && (6.68)\\
D_{15} &= \tilde{C}_{15}^{\lll 21} - C_{15}^{\lll 21} && (6.69)\\
\tilde{D}_{15} &= D_{15} - G(\tilde{A}_{19}, \tilde{B}_{15}, \tilde{C}_{15}) + G(A_{19}, B_{15}, C_{15}) && (6.70)\\
C_{11} &= \tilde{D}_{15}^{\lll 25} - D_{15}^{\lll 25}. && (6.71)
\end{aligned}$$

The solutions obtained for the above equations should also satisfy the following two conditions.

$$\begin{aligned}
G(\tilde{B}_{15}, \tilde{C}_{15}, \tilde{D}_{15}) - G(B_{15}, C_{15}, D_{15}) &= 1 && (6.72)\\
F(\tilde{C}_{15}, \tilde{D}_{15}, \tilde{A}_{15}) - F(C_{15}, D_{15}, A_{15}) &= \tilde{B}_{15}^{\lll 13} - B_{15}^{\lll 13}. && (6.73)
\end{aligned}$$

In his paper Dobbertin suggests an algorithm to solve the set of non-linear equations described by (6.67) to (6.73). This algorithm is replicated below.

Algorithm 6.2 *Dobbertin's Algorithm for Producing Almost-Inner Collisions*

1. *Choose A_{19}, B_{19}, C_{19}, D_{19}, B_{15} and C_{15} randomly. Compute $\tilde{B}_{19}$, $\tilde{C}_{19}$, $\tilde{B}_{15}$, $\tilde{C}_{15}$, D_{15} and $\tilde{D}_{15}$ as described in (6.67) to (6.71). Test if (6.72) is satisfied. If the test is passed goto 2.*

2. *Take A_{19}, B_{19}, C_{19}, D_{19}, B_{15} and C_{15} found in 1 as "basic values". Change one random bit in each of these variables, compute $\tilde{B}_{19}$, $\tilde{C}_{19}$, $\tilde{B}_{15}$, $\tilde{C}_{15}$, D_{15} and $\tilde{D}_{15}$ and test if (6.72) is satisfied. Then test if the left 4 bits of (6.73) are equal to 0. If this test is passed take the corresponding values A_{19}, B_{19}, C_{19}, D_{19}, B_{15} and C_{15} as the new "basic values". The next is doing the same as before, but now testing if the 8 left bits of (6.73) instead of 4 bits are zero. Continue with the left 12,16 . . . left bits until (6.73) is fulfilled.*

3. *Now (6.72) and (6.73) are satisfied and we obtain an inner almost-collision by setting $B_{11} = 0$ and defining A_{11}, C_{11}, D_{11} and $X[i]$ according to equations (6.58) to (6.66).*

For the inner almost-collision to be admissible, it is required that the following equation holds:

$$G(\tilde{B}_{19}, \tilde{C}_{19}, \tilde{D}_{19}) = G(B_{19}, C_{19}, D_{19}) \tag{6.74}$$

If equation (6.74) does not hold the differential attack is unlikely to succeed. Algorithm 6.2 has to be repeated until equation 6.74 is satisfied.

In the next section an alternative algorithm for solving the above set of non-linear Boolean equations is presented.

6.8 ALTERNATIVE ALGORITHM FOR ESTABLISHING INNER ALMOST-COLLISIONS

In this section we proceed to describe an alternative algorithm that leads to the solution of equations (6.50) to (6.57) and the establishment of inner almost-collisions.

For equations (6.50) to (6.57), make the following settings.

$$\begin{aligned} B_{11} &= \texttt{0x00000000} & A_{15} &= \texttt{0xFFFFFFFF} \\ C_{11} &= \texttt{0x00000000} & \tilde{A}_{15} &= \texttt{0x00000000} \end{aligned}$$

The choice of $\tilde{A}_{15}$ and A_{15} immediately satisfies equation (6.50). The choices for B_{11} and C_{11} implies that D_{15} and $\tilde{D}_{15}$ are equal. A collision can now be established by setting:

$$\begin{aligned} D_{15} &= \texttt{0xFFFDFFFE} \\ \tilde{D}_{15} &= D_{15} \\ C_{15} &= \texttt{0xEDFFCFFF} \\ \tilde{C}_{15} &= \texttt{0xFDFFDFFF} \\ \tilde{B}_{15} &= B_{15} + \tilde{B}_{19}^{\lll 19} - B_{19}^{\lll 19} - 1 \end{aligned}$$

The values for $\tilde{D}_{15}$, D_{15}, $\tilde{C}_{15}$ and C_{15} are chosen to satisfy equations (6.52) and (6.53) and to facilitate the easy manipulation of the functions $F(X,Y,Z)$ and $G(X,Y,Z)$. The choice of the relationship between $\tilde{B}_{15}$ and B_{15} ensures that it is easy to find a solution to equation (6.57). The following set of equations now needs to be solved.

$$\begin{aligned} F(\tilde{C}_{15},\tilde{D}_{15},\tilde{A}_{15}) - F(C_{15},D_{15},A_{15}) &= \tilde{B}_{15}^{\lll 13} - B_{15}^{\lll 13} && (6.75) \\ G(\tilde{B}_{15},\tilde{C}_{15},\tilde{D}_{15}) - G(B_{15},C_{15},D_{15}) &= \texttt{0xFFFFFFFF} && (6.76) \\ G(\tilde{A}_{19},\tilde{B}_{15},\tilde{C}_{15}) - G(A_{19},B_{15},C_{15}) &= 0 && (6.77) \\ G(\tilde{D}_{19},\tilde{A}_{19},\tilde{B}_{15}) - G(D_{19},A_{19},B_{15}) &= C_{15} - \tilde{C}_{15} + \\ &\quad \tilde{C}_{19}^{\lll 23} - C_{19}^{\lll 23} && (6.78) \\ G(\tilde{C}_{19},\tilde{D}_{19},\tilde{A}_{19}) - G(C_{19},D_{19},A_{19}) &= 0 && (6.79) \end{aligned}$$

Thus equations (6.50) to (6.57) can be reduced to equations (6.75) to (6.79). Note that equations (6.75) to (6.79) each contain a single unknown variable. It is now possible to define an algorithm that has a high probability to yield an admissible inner almost-collision. The suggested algorithm for finding an inner almost-collision is defined below:

Algorithm 6.3 *Alternative Algorithm for Producing inner almost-Collisions*

1. *Choose a random value for $\tilde{B}_{15}$ and compute B_{15}. Repeat this step until equation (6.75) and (6.76) are satisfied.*

2. *Choose a random value for A_{19}. Repeat this step until equation (6.77) is satisfied.*

3. *Choose a random value for D_{19}, $\tilde{B}_{19}$, $\tilde{C}_{19}$ and compute B_{19} and C_{19}. Repeat this step until equation (6.78) and equation (6.79) are satisfied.*

4. *Verify if equation (6.74) holds.*

   (a) *If equation (6.74) holds an admissible inner collision was found. Proceed to construct M_1 and M_2 as described by equations (6.58) to (6.66).*

   (b) *If equation (6.74) does not hold, repeat this algorithm from step 1.*

Once an admissible inner almost-collision is found, the differential attack described in [14] may be used to find a collision for all three rounds of MD4.

6.9 RESULTS

When comparing the performance of Algorithm 6.2 with that of Algorithm 6.3, two observations are made.

6.9.1 Number of Collisions

It is noted that when Algorithm 6.3 is used to find an admissible inner almost-collision, only a subset of all possible admissible inner almost-collisions is produced. This is due to the selection of A_{15}, $\tilde{A}_{15}$, B_{15}, $\tilde{B}_{15}$, C_{15} and $\tilde{C}_{15}$. When constructing an admissible inner almost-collision the attacker is free to choose 5 variables at random. This leaves the attacker with 2^{160} options. Each random choice does not however guarantee an admissible inner almost-collision. On average the attacker has to make 2^8 random choices for the 5 variables before an inner almost-collision is found. This reduces the number of inner almost-collisions to an estimated 2^{152}. According to [14] approximately one in every nine inner almost-collisions are admissible. Thus approximately 2^{149} admissible inner almost-collisions can be found with Algorithm 6.3. It is pointed out in [14] that it is possible to construct approximately 2^{106} message pairs that yield a collision for each admissible inner almost-collision under the MD4 hash function. Thus with the use of Algorithm 6.3 it is possible, to generate approximately 2^{255} message pairs that hash to the same value using MD4. When using Algorithm 6.2 the number of message pairs that result in a collision are estimated at 2^{281}.

6.9.2 Speedup Factor

Algorithm 6.3 has an advantage over Algorithm 6.2 when the number of operations required to find an admissible inner almost-collision is considered. A practical measurement of Algorithm 6.2 shows that approximately $O(2^{14})$ trials are required to find an admissible inner almost-collision. A similar measurement shows that Algorithm 6.3 requires on average $O(2^{8})$ trials to find an admissible inner almost-collision. This represents a speedup factor of approximately 64.

6.9.3 Example

An example of two messages that were constructed using Algorithm 6.3 for finding admissible inner almost-collisions is shown below. The common hash value is included for reference.

$$\begin{aligned}
X[0] &= \texttt{0xD6E3C2EA} & X[8] &= \texttt{0x25B0C32D}\\
X[1] &= \texttt{0x31759BA4} & X[9] &= \texttt{0xD1E9E09B}\\
X[2] &= \texttt{0x09028A49} & X[10] &= \texttt{0xEC08A64A}\\
X[3] &= \texttt{0x00DC9F7B} & X[11] &= \texttt{0x32CC035A}\\
X[4] &= \texttt{0x9688334C} & X[12] &= \texttt{0x669080A4}\\
X[5] &= \texttt{0x6A848F6B} & X[13] &= \texttt{0x31C4794B}\\
X[6] &= \texttt{0xB5E292DD} & X[14] &= \texttt{0xFFFFBFFB}\\
X[7] &= \texttt{0x4DCC5516} & X[15] &= \texttt{0xA281EB3F}
\end{aligned}$$

$$\tilde{X}[12] = \texttt{0x669080A5}$$

Common Digest:

```
0x94a568a0 0x84678967 0xea8da2b9 0x055dd5ab
```

6.10 CONCLUSION

This chapter contains a concise description of the operation of MD4. Attention has been paid in particular to the three rounds that constitutes the compress function for MD4. This is due to the importance of these rounds in the cryptanalysis of MD4 as presented in this Chapter. An implementation of the MD4 algorithm is attached as Appendix B.

The description of MD4 is followed by a description of the attack by Dobbertin on MD4. It is shown that a speedup of the attack on MD4 is possible. The speedup factor is estimated to be a factor of 64. The improvement in speed is attained at the cost of a reduction in the number of possible messages which result in a collision. These results were also presented at the Comsig 97 conference [56].

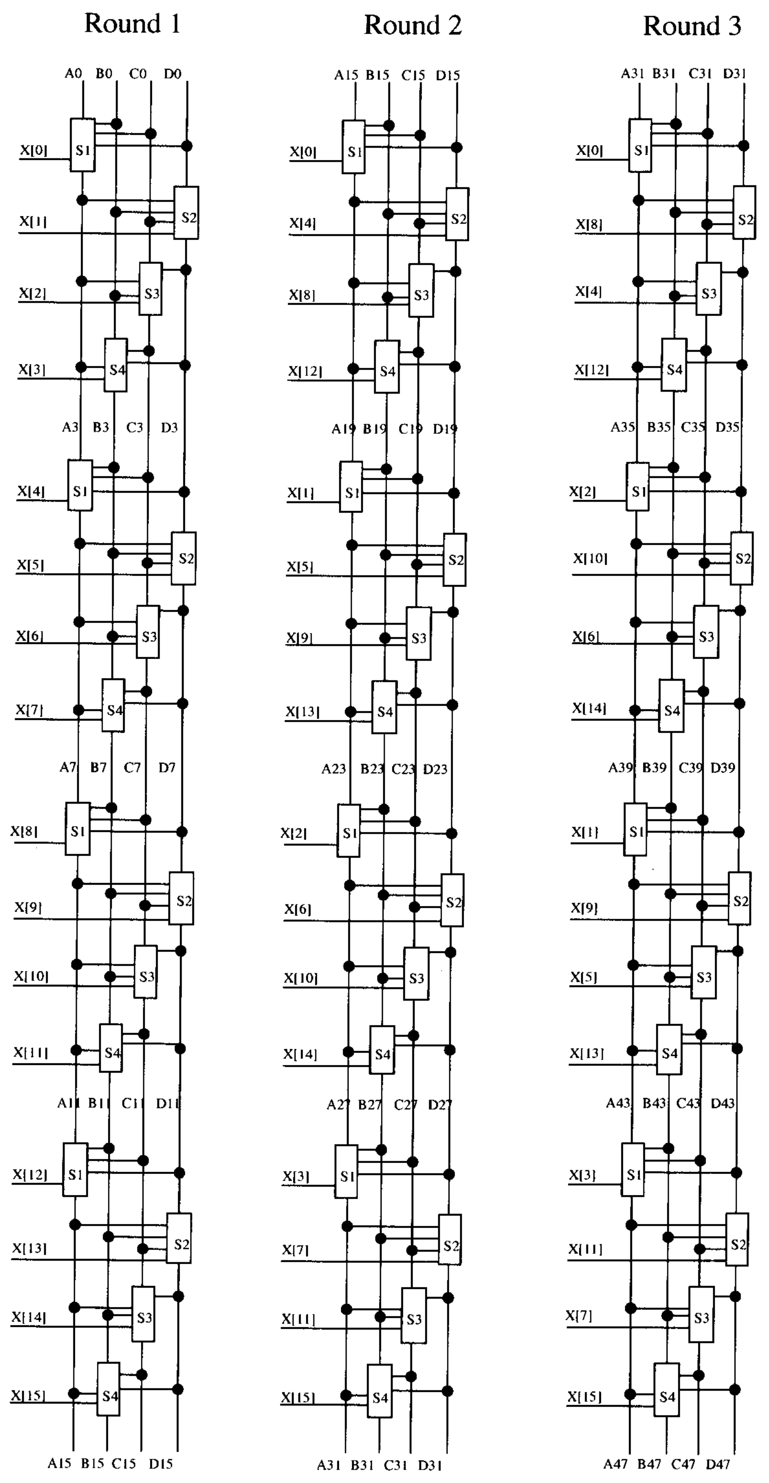

Round 1
Round 2
Round 3
A0 B0 C0 D0
A15 B15 C15 D15
A31 B31 C31 D31
S1
S2
S3
S4
X[0] X[1] X[2] X[3] X[4] X[5] X[6] X[7] X[8] X[9] X[10] X[11] X[12] X[13] X[14] X[15]
X[0] X[4] X[8] X[12] X[1] X[5] X[9] X[13] X[2] X[6] X[10] X[14] X[3] X[7] X[11] X[15]
X[0] X[8] X[4] X[12] X[2] X[10] X[6] X[14] X[1] X[9] X[5] X[13] X[3] X[11] X[7] X[15]
A3 B3 C3 D3
A7 B7 C7 D7
A11 B11 C11 D11
A19 B19 C19 D19
A23 B23 C23 D23
A27 B27 C27 D27
A35 B35 C35 D35
A39 B39 C39 D39
A43 B43 C43 D43
A15 B15 C15 D15
A31 B31 C31 D31
A47 B47 C47 D47

Figure 6.3: Diagrammatic Representation of the Compress Function of MD4

CHAPTER 7: ANALYSIS OF THE MD5 HASH ALGORITHM

7.1 INTRODUCTION

In this chapter we begin with a concise description of the MD5 hash algorithm. We then proceed by reconstructing the attack on MD5 as formulated by Dobbertin. The reconstruction is based on the source code used by Dobbertin to implement the attack. This is the first time a detailed description of the attack on MD5 is published.

7.2 INTRODUCTION TO MD5

MD5 is a dedicated hash function proposed by R. Rivest [45]. MD5 is the successor to MD4. MD5 is an extension of MD4. In extending MD4 to become MD5, a more conservative approach was taken. MD5 has the following features not encountered in MD4:

1. A fourth round.
2. A unique additive constant for each round.
3. The result of each step is added in the following step.
4. The permutations on the message words for rounds 2 and 3 were changed.
5. The rotation factors for each round were optimised for maximum avalanche effect.
6. The rotation factors for each round is unique.
7. The elementary function used in the second round was changed from $(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$ to $(X \wedge Z) \vee (Y \wedge (\neg Z))$. The intention of this change is to reduce the symmetry in $g()$.

The most prominent design criterion is security. In [45] it is conjectured that it is computationally infeasible to find two messages, M_1 and M_2, that hashes to the same value, or to find a message that results in a specified hash value. In other words, MD5 is intended to be both collision resistant and pre-image resistant.

A complete definition of MD5, including the padding rule is given in [45]. In view of the analysis of MD5 it is useful to consider the operation of the algorithm.

7.3 NOTATION

Before proceeding to describe the operation of MD5 it is convenient to introduce the following notation:

$$\begin{aligned}
X_j &= \text{32-bit word,} \qquad j \in \{0,1,2\ldots 15\} \\
\tilde{X}_j &= \text{Alternative 32-bit word,} \quad j \in \{0,1,2,\ldots 15\} \\
(AA,BB,CC,DD) &= \text{Hash variables} \\
(A_i,B_i,C_i,D_i) &= \text{Chaining variables after step } i, \quad i \in \{0,1,2,\ldots 47\}
\end{aligned}$$

7.4 THE MD5 ALGORITHM

Algorithm 7.1 *MD5 hash algorithm*

1. *Pad message.*
2. *Append the length of the message.*
3. *Hash and chaining variable initialisation.*
4. *Round 1.*
5. *Round 2.*
6. *Round 3.*
7. *Round 4.*
8. *Update hash variables.*
9. *Has the entire message been processed ?*
   (a) *No: Repeat from step 4.*
   (b) *Yes: Continue.*
10. *Output hash value.*

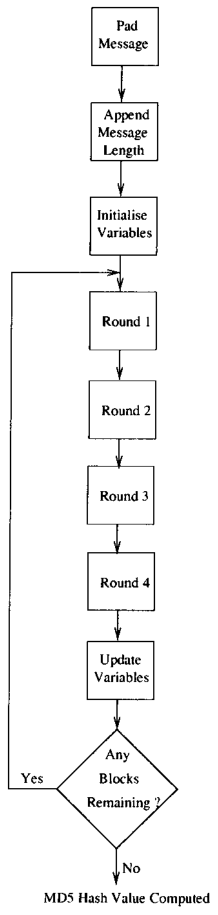


Figure 7.1: MD5 Block Diagram

MD5 is an iterated hash function based on the Dåmgard-Merkle construction [22], [23]. Each iteration requires the application of the compress function. The MD5 compress function is defined by the sequential application of four distinct rounds. The elementary size of a message block is 512 bits. If the message is not a multiple of 512 bits, a padding rule is used. Before the message block is processed, it is divided into 16 blocks of 32 bits each. Four 32-bit chaining variables are used, producing a 128-bit hash value. Algorithm 7.1 presents the main steps in MD5 along with a block diagrammatic representation of the structure of MD5 (see Figure 7.1). A description of each of the steps identified in the MD4 algorithm is presented next.

7.4.1 Message Padding

The first two steps ensure that the message length is a multiple of 512 bits. This allows the message to be processed in blocks of 512 bits at a time. The padding rule is described in [45].

7.4.2 Initial Values

The four chaining variables are initialised as:

$$\begin{aligned} A_0 &= \texttt{0x67452301} \\ B_0 &= \texttt{0xEFCDAB89} \\ C_0 &= \texttt{0x98BADCFE} \\ D_0 &= \texttt{0x10325476}. \end{aligned}$$

The hash variables contains the hash value for each iteration and is initialised as shown below:

$$\begin{aligned} AA &= A_0 \\ BB &= B_0 \\ CC &= C_0 \\ DD &= D_0. \end{aligned}$$

Note that this is identical to the initial values used for MD4 [10], [44].

7.4.3 Iterative Rounds

Steps 4-9 perform the iterative computation of the hash value. The hash value is obtained from the application of four distinct rounds to each 512 bit block of the message. The hash function derives its strength from these four rounds. The hash variables are updated once all four rounds have been completed. If all of the 512 bit blocks have been processed the updated hash variables contain the final hash value. The elementary operation within each round is described by:

$$\begin{aligned} R(S,T,U,V,X,K,W,r,j,i) &= S + (f_r(T,U,V) + X_j + T_i) \lll W_{r_j} \\ S &= R(S,T,U,V,X,K,W,r,j) \end{aligned}$$

with:

$$\begin{aligned} S,T,U,V &= \text{32 bit words} \\ X_j &= j\text{'th 32 bit word of the message,} \quad j \in \{0,1,2\ldots15\} \\ r &= \text{Round } r \quad r \in \{1,2,3,4\} \\ i &= \text{Step } (r-1)\cdot 16 + j \\ T_i &= \text{The i'th constant} \\ f_r &= \text{The boolean function in the r'th round} \\ x \lll W_{r_j} &= \text{Circular rotate } x \text{ left by} W_{r_j} \text{ bits} \\ x + y &= \text{Modulo } 2^{32} \text{ addition of } x \text{ and } y. \end{aligned}$$

Each round differs from the other with regard f_r. The index j in X_j is used to permutate the 512 bit input in 32-bit blocks for each round of the hash function. In each round W takes on one of four values. For each step the additive constant T_i is unique.

Four Boolean functions are defined for MD4.

$$\begin{aligned} f(X,Y,Z) &= (X \wedge Y) \vee ((\neg X) \wedge Z) \\ g(X,Y,Z) &= (X \wedge Z) \vee (Y \wedge (\neg Z)) \\ h(X,Y,Z) &= X \oplus Y \oplus Z \\ i(X,Y,Z) &= Y \oplus (X \vee (\neg Z)). \end{aligned}$$

with:

$\wedge$	$=$	Bitwise AND	$\neg$	$=$	Bitwise NOT
$\vee$	$=$	Bitwise OR	$\oplus$	$=$	Bitwise XOR.

The constants for T_i are defined in [45] and may be found in Appendix D which contains the source code for an implementation of MD5.

A graphical representation of a single step in the MD5 round function is shown in Figure 7.2

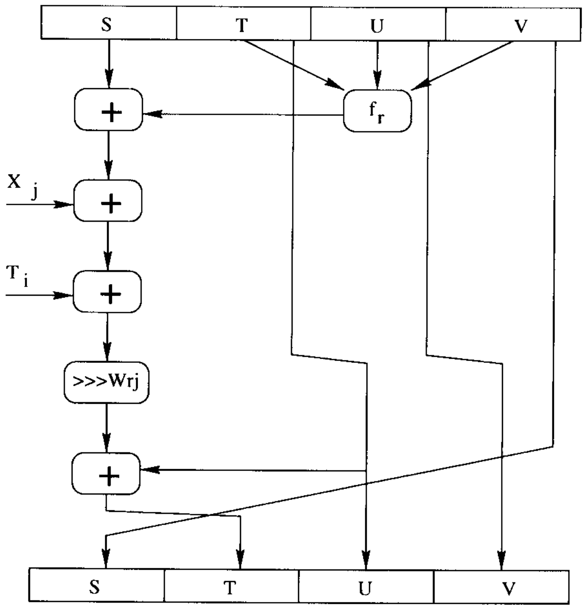


Figure 7.2: A Single Step in the MD5 Round Function

Four distinct rounds constitute the compress function for MD5. The following equations describe these rounds.

Round 1

The rotation constants for the first round are defined as:

$$fs1 = 7 \qquad fs3 = 17$$
$$fs2 = 12 \qquad fs4 = 22$$

The set of equations describing the first round are shown below:

$$A_3 = B_0 + (A_0 + f(B_0, C_0, D_0) + X_0 + T_0)^{\lll fs1} \tag{7.1}$$

$$D_3 = A_3 + (D_0 + f(A_3, B_0, C_0) + X_1 + T_1)^{\lll fs2} \tag{7.2}$$

$$C_3 = D_3 + (C_0 + f(D_3, A_3, B_0) + X_2 + T_2)^{\lll fs3} \tag{7.3}$$

$$B_3 = C_3 + (B_0 + f(C_3, D_3, A_3) + X_3 + T_3)^{\lll fs4} \tag{7.4}$$

$$A_7 = B_3 + (A_3 + f(B_3, C_3, D_3) + X_4 + T_4)^{\lll fs1} \tag{7.5}$$

$$D_7 = A_7 + (D_3 + f(A_7, B_3, C_3) + X_5 + T_5)^{\lll fs2} \tag{7.6}$$

$$C_7 = D_7 + (C_3 + f(D_7, A_7, B_3) + X_6 + T_6)^{\lll fs3} \tag{7.7}$$

$$B_7 = C_7 + (B_3 + f(C_7, D_7, A_7) + X_7 + T_7)^{\lll fs4} \tag{7.8}$$

$$A_{11} = B_7 + (A_7 + f(B_7, C_7, D_7) + X_8 + T_8)^{\lll fs1} \tag{7.9}$$

$$D_{11} = A_{11} + (D_7 + f(A_{11}, B_7, C_7) + X_9 + T_9)^{\lll fs2} \tag{7.10}$$

$$C_{11} = D_{11} + (C_7 + f(D_{11}, A_{11}, B_7) + X_{10} + T_{10})^{\lll fs3} \tag{7.11}$$

$$B_{11} = C_{11} + (B_7 + f(C_{11}, D_{11}, A_{11}) + X_{11} + T_{11})^{\lll fs4} \tag{7.12}$$

$$A_{15} = B_{11} + (A_{11} + f(B_{11}, C_{11}, D_{11}) + X_{12} + T_{12})^{\lll fs1} \tag{7.13}$$

$$D_{15} = A_{15} + (D_{11} + f(A_{15}, B_{11}, C_{11}) + X_{13} + T_{13})^{\lll fs2} \tag{7.14}$$

$$C_{15} = D_{15} + (C_{11} + f(D_{15}, A_{15}, B_{11}) + X_{14} + T_{14})^{\lll fs3} \tag{7.15}$$

$$B_{15} = C_{15} + (B_{11} + f(C_{15}, D_{15}, A_{15}) + X_{15} + T_{15})^{\lll fs4} \tag{7.16}$$

Round 2

The rotation constants used in round two are defined as:

$$gs1 = 5 \qquad gs3 = 14$$

$$gs2 = 9 \qquad gs4 = 20$$

The second round in the compress function is described by:

$$A_{19} = B_{15} + (A_{15} + g(B_{15}, C_{15}, D_{15}) + X_1 + T_{16})^{\lll gs1} \tag{7.17}$$
$$D_{19} = A_{19} + (D_{15} + g(A_{19}, B_{15}, C_{15}) + X_6 + T_{17})^{\lll gs2} \tag{7.18}$$
$$C_{19} = D_{19} + (C_{15} + g(D_{19}, A_{19}, B_{15}) + X_{11} + T_{18})^{\lll gs3} \tag{7.19}$$
$$B_{19} = C_{19} + (B_{15} + g(C_{19}, D_{19}, A_{19}) + X_0 + T_{19})^{\lll gs4} \tag{7.20}$$
$$A_{23} = B_{19} + (A_{19} + g(B_{19}, C_{19}, D_{19}) + X_5 + T_{20})^{\lll gs1} \tag{7.21}$$
$$D_{23} = A_{23} + (D_{19} + g(A_{23}, B_{19}, C_{19}) + X_{10} + T_{21})^{\lll gs2} \tag{7.22}$$
$$C_{23} = D_{23} + (C_{19} + g(D_{23}, A_{23}, B_{19}) + X_{15} + T_{22})^{\lll gs3} \tag{7.23}$$
$$B_{23} = C_{23} + (B_{19} + g(C_{23}, D_{23}, A_{23}) + X_4 + T_{23})^{\lll gs4} \tag{7.24}$$
$$A_{27} = B_{23} + (A_{23} + g(B_{23}, C_{23}, D_{23}) + X_9 + T_{24})^{\lll gs1} \tag{7.25}$$
$$D_{27} = A_{27} + (D_{23} + g(A_{27}, B_{23}, C_{23}) + X_{14} + T_{25})^{\lll gs2} \tag{7.26}$$
$$C_{27} = D_{27} + (C_{23} + g(D_{27}, A_{27}, B_{23}) + X_3 + T_{26})^{\lll gs3} \tag{7.27}$$
$$B_{27} = C_{27} + (B_{23} + g(C_{27}, D_{27}, A_{27}) + X_8 + T_{27})^{\lll gs4} \tag{7.28}$$
$$A_{31} = B_{27} + (A_{27} + g(B_{27}, C_{27}, D_{27}) + X_{13} + T_{28})^{\lll gs1} \tag{7.29}$$
$$D_{31} = A_{31} + (D_{27} + g(A_{31}, B_{27}, C_{27}) + X_2 + T_{29})^{\lll gs2} \tag{7.30}$$
$$C_{31} = D_{31} + (C_{27} + g(D_{31}, A_{31}, B_{27}) + X_7 + T_{30})^{\lll gs3} \tag{7.31}$$
$$B_{31} = C_{31} + (B_{27} + g(C_{31}, D_{31}, A_{31}) + X_{12} + T_{31})^{\lll gs4} \tag{7.32}$$

Round 3

The rotation constants used in the third round are defined as:

$$hs1 = 4 \qquad hs3 = 16$$
$$hs2 = 11 \qquad hs4 = 23$$

The third round of the MD5 compress function is defined by:

$$\begin{aligned}
A_{35} &= B_{31} + (A_{31} + h(B_{31}, C_{31}, D_{31}) + X_5 + T_{32})^{\lll hs1} && (7.33)\\
D_{35} &= A_{35} + (D_{31} + h(A_{35}, B_{31}, C_{31}) + X_8 + T_{33})^{\lll hs2} && (7.34)\\
C_{35} &= D_{35} + (C_{31} + h(D_{35}, A_{35}, B_{31}) + X_{11} + T_{34})^{\lll hs3} && (7.35)\\
B_{35} &= C_{35} + (B_{31} + h(C_{35}, D_{35}, A_{35}) + X_{14} + T_{35})^{\lll hs4} && (7.36)\\
A_{39} &= B_{35} + (A_{35} + h(B_{35}, C_{35}, D_{35}) + X_1 + T_{36})^{\lll hs1} && (7.37)\\
D_{39} &= A_{39} + (D_{35} + h(A_{39}, B_{35}, C_{35}) + X_4 + T_{37})^{\lll hs2} && (7.38)\\
C_{39} &= D_{39} + (C_{35} + h(D_{39}, A_{39}, B_{35}) + X_7 + T_{38})^{\lll hs3} && (7.39)\\
B_{39} &= C_{39} + (B_{35} + h(C_{39}, D_{39}, A_{39}) + X_{10} + T_{39})^{\lll hs4} && (7.40)\\
A_{43} &= B_{39} + (A_{39} + h(B_{39}, C_{39}, D_{39}) + X_{13} + T_{40})^{\lll hs1} && (7.41)\\
D_{43} &= A_{43} + (D_{39} + h(A_{43}, B_{39}, C_{39}) + X_0 + T_{41})^{\lll hs2} && (7.42)\\
C_{43} &= D_{43} + (C_{39} + h(D_{43}, A_{43}, B_{39}) + X_3 + T_{42})^{\lll hs3} && (7.43)\\
B_{43} &= C_{43} + (B_{39} + h(C_{43}, D_{43}, A_{43}) + X_6 + T_{43})^{\lll hs4} && (7.44)\\
A_{47} &= B_{43} + (A_{43} + h(B_{43}, C_{43}, D_{43}) + X_9 + T_{44})^{\lll hs1} && (7.45)\\
D_{47} &= A_{47} + (D_{43} + h(A_{47}, B_{43}, C_{43}) + X_{12} + T_{45})^{\lll hs2} && (7.46)\\
C_{47} &= D_{47} + (C_{43} + h(D_{47}, A_{47}, B_{43}) + X_{15} + T_{46})^{\lll hs3} && (7.47)\\
B_{47} &= C_{47} + (B_{43} + h(C_{47}, D_{47}, A_{47}) + X_2 + T_{47})^{\lll hs4} && (7.48)
\end{aligned}$$

Round 4

The fourth round employs the following rotation constants:

$$\begin{aligned}
is1 &= 6 & \qquad is3 &= 15\\
ihs2 &= 10 & \qquad is4 &= 21
\end{aligned}$$

The final round in the MD5 compress function is obtained from:

$$A_{51} = B_{47} + (A_{47} + i(B_{47}, C_{47}, D_{47}) + X_0 + T_{48})^{\lll is1} \quad (7.49)$$
$$D_{51} = A_{51} + (D_{47} + i(A_{51}, B_{47}, C_{47}) + X_7 + T_{49})^{\lll is2} \quad (7.50)$$
$$C_{51} = D_{51} + (C_{47} + i(D_{51}, A_{51}, B_{47}) + X_{14} + T_{50})^{\lll is3} \quad (7.51)$$
$$B_{51} = C_{51} + (B_{47} + i(C_{51}, D_{51}, A_{51}) + X_5 + T_{51})^{\lll is4} \quad (7.52)$$
$$A_{55} = B_{51} + (A_{51} + i(B_{51}, C_{51}, D_{51}) + X_{12} + T_{52})^{\lll is1} \quad (7.53)$$
$$D_{55} = A_{55} + (D_{51} + i(A_{55}, B_{51}, C_{51}) + X_3 + T_{53})^{\lll is2} \quad (7.54)$$
$$C_{55} = D_{55} + (C_{51} + i(D_{55}, A_{55}, B_{51}) + X_{10} + T_{54})^{\lll is3} \quad (7.55)$$
$$B_{55} = C_{55} + (B_{51} + i(C_{55}, D_{55}, A_{55}) + X_1 + T_{55})^{\lll is4} \quad (7.56)$$
$$A_{59} = B_{55} + (A_{55} + i(B_{55}, C_{55}, D_{55}) + X_8 + T_{56})^{\lll is1} \quad (7.57)$$
$$D_{59} = A_{59} + (D_{55} + i(A_{59}, B_{55}, C_{55}) + X_{15} + T_{57})^{\lll is2} \quad (7.58)$$
$$C_{59} = D_{59} + (C_{55} + i(D_{59}, A_{59}, B_{55}) + X_6 + T_{58})^{\lll is3} \quad (7.59)$$
$$B_{59} = C_{59} + (B_{55} + i(C_{59}, D_{59}, A_{59}) + X_{13} + T_{59})^{\lll is4} \quad (7.60)$$
$$A_{63} = B_{59} + (A_{59} + i(B_{59}, C_{59}, D_{59}) + X_4 + T_{60})^{\lll is1} \quad (7.61)$$
$$D_{63} = A_{63} + (D_{59} + i(A_{63}, B_{59}, C_{59}) + X_{11} + T_{61})^{\lll is2} \quad (7.62)$$
$$C_{63} = D_{63} + (C_{59} + i(D_{63}, A_{63}, B_{59}) + X_2 + T_{62})^{\lll is3} \quad (7.63)$$
$$B_{63} = C_{63} + (B_{59} + i(C_{63}, D_{63}, A_{63}) + X_9 + T_{63})^{\lll is4} \quad (7.64)$$

Update Variables

After completion of all three rounds, the hash variables are updated as shown below:

$$AA = A_0 = A_{47} + AA$$
$$BB = B_0 = B_{47} + BB$$
$$CC = C_0 = C_{47} + CC$$
$$DD = D_0 = D_{47} + DD$$

Once the last 512 bit message block have been processed, the final hash value is given by AA, BB, CC, and DD. If there are unprocessed message blocks remaining, A_0, B_0, C_0, and D_0 contains the new initial values for the next iteration of MD5.

7.5 ANALYSIS OF MD5

MD5 is a dedicated hash function as described in Section 7.4. As noted in Section 7.4 MD5 is an extension of MD4. MD5 was designed to be a more secure hash function and is therefore more conservative in design than MD4. In 1996 Dobbertin presented an attack on MD4, showing that it is possible to find collisions for MD4 in less than a second on a personal computer. The cryptanalysis of MD4 is presented in [14]. Additional work relating to the cryptanalysis of MD4 is presented in [17]. Dobbertin applied similar cryptanalytical techniques to RIPEMD and found that the first two rounds and last two rounds of RIPEMD are not collision resistant [54]. At the rump session of EUROCRYPT'96 it was announced that it is possible to find collisions for the compress function of MD5 [54]. An outline of this attack was published in [55]. In [12] it is stated that using these techniques, the attack requires approximately 10 hours on a personal computer with a Pentium processor. From these publications it appears as if the techniques employed in the cryptanalysis of MD5 are similar to those used on MD4 and RIPEMD.

The attack on MD5 as described in this chapter is based mostly on the notes of Antoon Bosselaers and the C source code developed by Dobbertin.

7.5.1 Notation

Before proceeding it is useful to introduce the following notation. Let

$$Z = \text{Message word, chaining variable or a collection of chaining variables.}$$
$$\tilde{Z} = \text{Alternative value for } Z.$$

In addition, the following operator is defined:

$$Z^{\ggg Y} = \text{Circular rotation of } Z \text{ to the right by } Y \text{ bits.}$$

and

$$-Z^{\ggg Y} = -(Z^{\ggg Y}) \text{ bits.}$$

Thus, rotation has precedence over negation.

7.5.2 Outline of the Attack

The attack on MD5 is based on the assumption that all the message words are identical except for one message word X_i. The difference between X_i and $\tilde{X}_i$ is given by:

$$\tilde{X}_i = X_i + \Delta \tag{7.65}$$

where Δ is a 32 bit word with a small Hamming weight. For the attack on MD5 as described by Dobbertin in [12] and [55], the following choices are made:

$$\Delta = \texttt{0x00000200} \tag{7.66}$$
$$X_i = X_{14}. \tag{7.67}$$

It is stated in [55] that it may be possible to utilise other message words as well as other values for Δ. The message word X_{14} is used in equations (7.15) (7.26), (7.36) and (7.51) (once in each round). Using the notation defined in Section 7.5.1 the following definitions are presented.

$COMPRESS_x^y$ = Value of chaining variables after equations y to x of the compress function was applied to the message.

$\widetilde{COMPRESS}_x^y$ = Value of chaining variables after equations y to x of the compress function was applied to the modified message.

The attack is now reduced to find two messages so that:

$$COMPRESS_{7.26}^{7.15} - \widetilde{COMPRESS}_{7.26}^{7.15} = 0 \tag{7.68}$$

and

$$COMPRESS_{7.51}^{7.36} - \widetilde{COMPRESS}_{7.51}^{7.36} = 0 \tag{7.69}$$

If these conditions are met, so-called inner collisions are established. The attacker then has to find suitable message words which would link equation (7.26) to equation (7.36). From the above it appears that the attack on MD5 is a divide and conquer attack. Three phases are identified in this attack.

1. Find a message such that an inner collision is established for steps (7.15) to (7.26).
2. Find a message such that an inner collision is established for steps (7.36) to (7.51).
3. Find suitable message words to link the results obtained for the first two phases.

The inner collisions may be found by deriving two sets of difference equations (one for each step). These equations then has to be solved simultaneously due to the large overlap in words. At the same time a link between these equations has to be sought.

Each of these steps are discussed in a separate section in this chapter. Reference implementations are attached as appendices. An efficient technique exists to determine if solutions to certain Boolean expressions of a particular form exists. This technique is common to the first two phases of the attack and is therefore discussed in a separate section. Use is also made of the continuous approximation techniques used in the analysis of MD4 and RIPEMD.

7.5.3 Phase I: Inner Collisons for First Two Rounds

The first phase deals with the solution of a set of difference equations with the specific aim of finding an inner collision, such that equation (7.68) is satisfied. The difference equations are derived from equations (7.15) to (7.26) and are written as:

$$
\begin{aligned}
(C_{15} - D_{15})^{\ggg fs3} - (\tilde{C}_{15} - D_{15})^{\ggg fs3} &= X_{14} - \tilde{X}_{14}. && (7.70)\\
(B_{15} - C_{15})^{\ggg fs4} - (\tilde{B}_{15} - \tilde{C}_{15})^{\ggg fs4} &= f(C_{15}, D_{15}, A_{15}) - f(\tilde{C}_{15}, D_{15}, A_{15}) && (7.71)\\
(A_{19} - B_{15})^{\ggg gs1} - (\tilde{A}_{19} - \tilde{B}_{15})^{\ggg gs1} &= g(B_{15}, C_{15}, D_{15}) - g(\tilde{B}_{15}, \tilde{C}_{15}, D_{15}) && (7.72)\\
(D_{19} - A_{19})^{\ggg gs2} - (\tilde{D}_{19} - \tilde{A}_{19})^{\ggg gs2} &= g(A_{19}, B_{15}, C_{15}) - g(\tilde{A}_{19}, \tilde{B}_{15}, \tilde{C}_{15}) && (7.73)\\
(C_{19} - D_{19})^{\ggg gs3} - (\tilde{C}_{19} - \tilde{D}_{19})^{\ggg gs3} &= C_{15} - \tilde{C}_{15} + g(D_{19}, A_{19}, B_{15}) - \\
&\quad\; g(\tilde{D}_{19}, \tilde{A}_{19}, \tilde{B}_{15}) && (7.74)\\
(B_{19} - C_{19})^{\ggg gs4} - (\tilde{B}_{19} - \tilde{C}_{19})^{\ggg gs4} &= B_{15} - \tilde{B}_{15} + g(C_{19}, D_{19}, A_{19}) - \\
&\quad\; g(\tilde{C}_{19}, \tilde{D}_{19}, \tilde{A}_{19}) && (7.75)
\end{aligned}
$$

$$\begin{aligned}
(A_{23} - B_{19})^{\ggg gs1} - (\tilde{A}_{23} - \tilde{B}_{19})^{\ggg gs1} &= A_{19} - \tilde{A}_{19} + g(B_{19}, C_{19}, D_{19}) - \\
&\quad g(\tilde{B}_{19}, \tilde{C}_{19}, \tilde{D}_{19}) && (7.76)\\
(D_{23} - A_{23})^{\ggg gs2} - (\tilde{D}_{23} - \tilde{A}_{23})^{\ggg gs2} &= D_{19} - \tilde{D}_{19} + g(A_{23}, B_{19}, C_{19}) - \\
&\quad g(\tilde{A}_{23}, \tilde{B}_{19}, \tilde{C}_{19}) && (7.77)\\
(C_{23} - D_{23})^{\ggg gs3} - (C_{23} - \tilde{D}_{23})^{\ggg gs3} &= C_{19} - \tilde{C}_{19} + g(D_{23}, A_{23}, B_{19}) - \\
&\quad g(\tilde{D}_{23}, \tilde{A}_{23}, \tilde{B}_{19}) && (7.78)\\
0 &= B_{19} - \tilde{B}_{19} + g(C_{23}, D_{23}, A_{23}) - \\
&\quad g(C_{23}, \tilde{D}_{23}, \tilde{A}_{23}) && (7.79)\\
0 &= A_{23} - \tilde{A}_{23} + g(B_{23}, C_{23}, D_{23}) - \\
&\quad g(B_{23}, C_{23}, \tilde{D}_{23}) && (7.80)\\
0 &= D_{23} - \tilde{D}_{23} + X_{14} - \tilde{X}_{14}. && (7.81)
\end{aligned}$$

Equations (7.70) to (7.81) may be simplified by making appropriate choices for certain chaining variables. From equation (7.81) the following condition is imposed on D_{23} and $\tilde{D}_{23}$:

$$\tilde{D}_{23} - D_{23} = X_{14} - \tilde{X}_{14} \tag{7.82}$$

from equation (7.66) equation (7.81) may be written as:

$$\tilde{D}_{23} - D_{23} = -1^{\lll 9}. \tag{7.83}$$

By setting $A_{23} = \tilde{A}_{23}$ and $B_{19} = \tilde{B}_{19}$ the conditions imposed by equations (7.80) and (7.79) are satisfied if:

$$g(B_{23}, C_{23}, \tilde{D}_{23}) = g(B_{23}, C_{23}, D_{23}) \tag{7.84}$$

and:

$$g(C_{23}, \tilde{D}_{23}, A_{23}) = g(C_{23}, D_{23}, A_{23}) \tag{7.85}$$

Equations (7.84) and (7.85) are satisfied relatively easily due to the low complexity and consequent ease with which the Boolean function $g()$ can be manipulated. If equations (7.84) and (7.85) holds, a high probability exists that:

$$g(D_{23}, A_{23}, B_{19}) = g(\tilde{D}_{23}, A_{23}, B_{19}).$$

If this is the case, equation (7.78) may be simplified to:

$$(C_{23} - D_{23})^{\ggg gs3} - (C_{23} - \tilde{D}_{23})^{\ggg gs3} = C_{19} - \tilde{C}_{19} \tag{7.86}$$

on the assumption that:

$$(C_{23} - D_{23})^{\ggg gs3} - (C_{23} - \tilde{D}_{23})^{\ggg gs3} = (\tilde{D}_{23} - D_{23})^{\ggg gs3}$$

The following condition is imposed on the relationship between C_{19} and $\tilde{C}_{19}$:

$$(\tilde{D}_{23} - D_{23})^{\ggg gs3} = C_{19} - \tilde{C}_{19}. \tag{7.87}$$

Thus:

$$C_{19} - \tilde{C}_{19} = -1^{\ggg 5}. \tag{7.88}$$

Given the above assumptions, the set of difference equations from (7.70) to (7.81) may be reduced to:

$$\begin{aligned}
(C_{15} - D_{15})^{\ggg fs3} - (\tilde{C}_{15} - D_{15})^{\ggg fs3} &= X_{14} - \tilde{X}_{14}. && (7.89)\\
(B_{15} - C_{15})^{\ggg fs4} - (\tilde{B}_{15} - \tilde{C}_{15})^{\ggg fs4} &= f(C_{15}, D_{15}, A_{15}) - f(\tilde{C}_{15}, D_{15}, A_{15}) && (7.90)\\
(A_{19} - B_{15})^{\ggg gs1} - (\tilde{A}_{19} - \tilde{B}_{15})^{\ggg gs1} &= g(B_{15}, C_{15}, D_{15}) - g(\tilde{B}_{15}, \tilde{C}_{15}, D_{15}) && (7.91)\\
(D_{19} - A_{19})^{\ggg gs2} - (\tilde{D}_{19} - \tilde{A}_{19})^{\ggg gs2} &= g(A_{19}, B_{15}, C_{15}) - g(\tilde{A}_{19}, \tilde{B}_{15}, \tilde{C}_{15}) && (7.92)\\
(C_{19} - D_{19})^{\ggg gs3} - (\tilde{C}_{19} - \tilde{D}_{19})^{\ggg gs3} &= C_{15} - \tilde{C}_{15} + g(D_{19}, A_{19}, B_{15}) - \\
&\quad g(\tilde{D}_{19}, \tilde{A}_{19}, \tilde{B}_{15}) && (7.93)\\
(B_{19} - C_{19})^{\ggg gs4} - (\tilde{B}_{19} - \tilde{C}_{19})^{\ggg gs4} &= B_{15} - \tilde{B}_{15} + g(C_{19}, D_{19}, A_{19}) - \\
&\quad g(\tilde{C}_{19}, \tilde{D}_{19}, \tilde{A}_{19}) && (7.94)\\
0 &= A_{19} - \tilde{A}_{19} + g(B_{19}, C_{19}, D_{19}) - \\
&\quad g(B_{19}, \tilde{C}_{19}, \tilde{D}_{19}) && (7.95)\\
(D_{23} - A_{23})^{\ggg gs2} - (\tilde{D}_{23} - A_{23})^{\ggg gs2} &= D_{19} - \tilde{D}_{19} + g(A_{23}, B_{19}, C_{19}) - \\
&\quad g(A_{23}, B_{19}, \tilde{C}_{19}). && (7.96)
\end{aligned}$$

Consider equation (7.89). By re-writing equation (7.89) as:

$$\tilde{C}_{15} = ((C_{15} - D_{15})^{\ggg fs3} + \tilde{X}_{14} - X_{14})^{\lll fs3} + D_{15}. \tag{7.97}$$

Valid solutions to C_{15}, $\tilde{C}_{15}$ and D_{15} may be found by setting:

$$C_{15} = D_{15} - 1 \tag{7.98}$$

substituting equations (7.98) and (7.66) in (7.97) the following expression for C_{15} is obtained.

$$\begin{aligned}
\tilde{C}_{15} &= ((-1)^{\ggg fs3} + 1^{\lll 9})^{\lll fs3} + D_{15} && (7.99)\\
&= (-1 + 1^{\lll 9})^{\lll fs3} + D_{15} && (7.100)\\
&= (-1 + 1^{\lll 9})^{\lll 17} + D_{15}. && (7.101)
\end{aligned}$$

With equations (7.98) and (7.101) in hand the difference between C_{15} and $\tilde{C}_{15}$ is obtained as:

$$\begin{aligned}
\tilde{C}_{15} - C_{15} &= ((-1 + 1^{\lll 9})^{\lll 17} + D_{15}) - (D_{15} - 1) && (7.102)\\
\tilde{C}_{15} - C_{15} &= ((-1 + 1^{\lll 9})^{\lll 17} + 1 && (7.103)\\
&= \texttt{0x03FE0001}. && (7.104)
\end{aligned}$$

By setting D_{15} to:

$$D_{15} = 1^{\lll 16} - 1^{\lll 25} - 1 \tag{7.105}$$

C_{15} and $\tilde{C}_{15}$ are easily computed as:

$$\begin{aligned}
C_{15} &= \texttt{0xFE00FFFF} && (7.106)\\
\tilde{C}_{15} &= \texttt{0x01FF0000}. && (7.107)
\end{aligned}$$

With equation (7.89) satisfied, further simplifications may be achieved by setting:

$$\begin{aligned}
D_{19} &= \texttt{0x00000000} && (7.108)\\
\tilde{D}_{19} &= \texttt{0xFFFFFFFF}. && (7.109)
\end{aligned}$$

These choices allow the simplification of equation (7.95) to:

$$\tilde{A}_{19} - A_{19} = C_{19} - B_{19}. \tag{7.110}$$

This simplification is a direct result of the manipulation of the bitwise Boolean function $g()$. With these simplifications the condition imposed by expression (7.68) may be written as:

$$COMPRESS^{7.15}_{7.22} - \widetilde{COMPRESS}^{7.15}_{7.22} = \{\varepsilon^1_1, \varepsilon^1_2, \varepsilon^1_3, \varepsilon^1_4\} \tag{7.111}$$

with:

$$\begin{aligned}
\varepsilon^1_1 &= \texttt{0x00000000} && (7.112)\\
\varepsilon^1_2 &= \texttt{0x00000000} && (7.113)\\
\varepsilon^1_3 &= \texttt{0x08000000} && (7.114)\\
\varepsilon^1_4 &= \texttt{0xF8000000}. && (7.115)
\end{aligned}$$

Given the above choices, equations (7.89) – (7.96) may be re-written as:

$$\begin{aligned}
0 &= X_{14} - \tilde{X}_{14} - (C_{15} - D_{15})^{\ggg fs3} + (\tilde{C}_{15} - D_{15})^{\ggg fs3}. && (7.116)\\
0 &= C_{15} - \tilde{C}_{15} + f(C_{15}, D_{15}, A_{15})^{\lll fs4} - f(\tilde{C}_{15}, D_{15}, A_{15})^{\lll fs4} - B_{15} + \tilde{B}_{15} && (7.117)\\
0 &= B_{15} - \tilde{B}_{15} + g(B_{15}, C_{15}, D_{15})^{\lll gs3} - g(\tilde{B}_{15}, \tilde{C}_{15}, D_{15})^{\lll gs3} - A_{19} + \tilde{A}_{19} && (7.118)\\
0 &= (D_{19} - A_{19})^{\ggg gs2} - (\tilde{D}_{19} - \tilde{A}_{19})^{\ggg gs2} - \\
&\quad g(A_{19}, B_{15}, C_{15}) + g(\tilde{A}_{19}, \tilde{B}_{15}, \tilde{C}_{15}) && (7.119)\\
0 &= (C_{19} - D_{19})^{\ggg gs3} - (\tilde{C}_{19} - \tilde{D}_{19})^{\ggg gs3} - C_{15} + \tilde{C}_{15} - \\
&\quad g(D_{19}, A_{19}, B_{15}) + g(\tilde{D}_{19}, \tilde{A}_{19}, \tilde{B}_{15}) && (7.120)\\
0 &= (A_{19} - \tilde{A}_{19})^{\ggg gs4} - (A_{19} - \tilde{A}_{19} - 1^{\lll 5})^{\ggg gs4} - B_{15} + \tilde{B}_{15} - \\
&\quad g(C_{19}, D_{19}, A_{19}) + g(\tilde{C}_{19}, \tilde{D}_{19}, \tilde{A}_{19}) && (7.121)\\
0 &= A_{19} - \tilde{A}_{19} + g(B_{19}, C_{19}, D_{19}) - g(B_{19}, \tilde{C}_{19}, \tilde{D}_{19}) && (7.122)\\
0 &= D_{19} - \tilde{D}_{19} + g(A_{23}, B_{19}, C_{19}) - g(A_{23}, B_{19}, \tilde{C}_{19}) - \\
&\quad (D_{23} - A_{23})^{\ggg gs2} + (\tilde{D}_{23} - A_{23})^{\ggg gs2}. && (7.123)
\end{aligned}$$

A solution to equations (7.89) to (7.96) will probably result in an inner collision for the first two rounds of MD5, given the assumptions and choices described earlier in this section. Algorithm 7.2 presents the procedure used by Dobbertin to find a solution to the set of difference equations as defined by equations (7.89) to (7.96).

Algorithm 7.2 *Construction of Inner Collision: Phase I*

1. *Make initial choices for* D_{15}, C_{15}, $\tilde{C}_{15}$, D_{19} *and* $\tilde{D}_{19}$ *as defined by equations (7.105) – (7.109). Set a counter* $n = 0$.

2. *Choose a random value for* A_{15}.

3. *Determine* $B_{15} - \tilde{B}_{15}$ *from equation (7.117).*

4. *Choose a random value for* B_{15} *and calculate* $\tilde{B}_{15}$ *from the result obtained in step 3. Proceed to step 6.*

5. *If* $n > 0$ *use the values for* $A_{15_{Basic}}$, $B_{15_{Basic}}$ *and* $C_{15_{Basic}}$ *as determined in step 14. Choose values with a small Hamming distance from these basic values for* A_{15}, B_{15} *and* C_{15} *and proceed to step 6.*

6. *Determine* $A_{19} - \tilde{A}_{19}$ *from equation (7.118).*

7. *Choose* C_{19} *at random and calculate* $\tilde{C}_{19}$ *according to equation (7.88).*

8. *From equation (7.121) determine all possible solutions for* A_{19}.

9. *Calculate all possible values for* $\tilde{A}_{19}$ *from the result obtained in step 6.*

10. *Determine, for each valid value of* A_{19} *and* $\tilde{A}_{19}$, *a possible solution to (7.119) by determining all possible valid values for* B_{15} *and* $\tilde{B}_{15}$ *(note that in future computations, these values for* B_{15} *and* $\tilde{B}_{15}$ *should be used, instead of the values computed in step 4).*

11. *Confirm whether equation (7.118) holds for the newly computed results obtained for* B_{15}, $\tilde{B}_{15}$, A_{19} *and* $\tilde{A}_{19}$.

12. *If equation (7.118) holds, confirm if equation (7.117) holds. If equation (7.118) does not hold, proceed to step 2.*

13. *If equation (7.117) holds, determine to which extent equation (7.120) holds.*

14. (a) *If the left* $4 \cdot n$ *bits of equation (7.120) are equal to zero, set* $n = n + 1$. *Preserve the current values for* A_{15}, B_{15} *and* C_{19} *as* $A_{15_{Basic}}$, $B_{15_{Basic}}$ *and* $C_{19_{Basic}}$.

    (b) *If* $0 < n < 8$ *and the left* $4 \cdot n$ *bits of equation (7.120) are not equal to zero, return to step 5.*

    (c) *If the left* $4 \cdot n$ *bits of equation (7.120) are not equal to zero, and* $n = 0$ *return to step 1.*

15. *If $n = 8$, determine if the assumptions made when reducing the set of difference equations from equations (7.70) – (7.81) to equations (7.116) – (7.123) holds. Confirm the validity of equations (7.78) – (7.81). Specifically confirm whether equation (7.123) holds. If all of these conditions are satisfied, there exists a high probability that an inner collision was found. If any of these conditions are not satisfied , return to step 1.*

Algorithm 7.2 may be modified to restart after a number of iterations to prevent dead ends, while searching for a solution. An algorithm which allows the construction of all possible solutions to an expression of a certain form is described in Section 7.5.6. Algorithm 7.2 produces an inner almost collision for the first two rounds of MD5 in less than one hour on a 120 MHz Pentium PC. An implementation of Algorithm 7.2 is attached as Appendix E.

7.5.4 Phase II: Inner Collisions for Last Two Rounds

The second phase is similar to the first phase insofar as it involves the solution of a set of difference equations, with the specific aim of finding an inner collision such that equation (7.69) is satisfied. The difference equations are derived from equations (7.36) to (7.51) and are written as:

$$(B_{35} - C_{35})^{\ggg hs4} - (\tilde{B}_{35} - C_{35})^{\ggg hs4} = X_{14} - \tilde{X}_{14} \tag{7.124}$$

$$(A_{39} - B_{35})^{\ggg hs1} - (\tilde{A}_{39} - \tilde{B}_{35})^{\ggg hs1} = h(B_{35}, C_{35}, D_{35}) - h(\tilde{B}_{35}, C_{35}, D_{35}) \tag{7.125}$$

$$(D_{39} - A_{39})^{\ggg hs2} - (\tilde{D}_{39} - \tilde{A}_{39})^{\ggg hs2} = h(A_{39}, B_{35}, C_{35}) - h(\tilde{A}_{39}, \tilde{B}_{35}, C_{35}) \tag{7.126}$$

$$(C_{39} - D_{39})^{\ggg hs3} - (\tilde{C}_{39} - \tilde{D}_{39})^{\ggg hs3} = h(D_{39}, A_{39}, B_{35}) - h(\tilde{D}_{39}, \tilde{A}_{39}, \tilde{B}_{35}) \tag{7.127}$$

$$(B_{39} - C_{39})^{\ggg hs4} - (\tilde{B}_{39} - \tilde{C}_{39})^{\ggg hs4} = B_{35} - \tilde{B}_{35} + h(C_{39}, D_{39}, A_{39}) - h(\tilde{C}_{39}, \tilde{D}_{39}, \tilde{A}_{39}) \tag{7.128}$$

$$(A_{43} - B_{39})^{\ggg hs1} - (\tilde{A}_{43} - \tilde{B}_{39})^{\ggg hs1} = A_{39} - \tilde{A}_{39} + h(B_{39}, C_{39}, D_{39}) - h(\tilde{B}_{39}, \tilde{C}_{39}, \tilde{D}_{39}) \tag{7.129}$$

$$(D_{43} - A_{43})^{\ggg hs2} - (\tilde{D}_{43} - \tilde{A}_{43})^{\ggg hs2} = D_{39} - \tilde{D}_{39} + h(A_{43}, B_{39}, C_{39}) - h(\tilde{A}_{43}, \tilde{B}_{39}, \tilde{C}_{39}) \tag{7.130}$$

$$(C_{43} - D_{43})^{\ggg hs3} - (\tilde{C}_{43} - \tilde{D}_{43})^{\ggg hs3} = C_{39} - \tilde{C}_{39} + h(D_{43}, A_{43}, B_{39}) - h(\tilde{D}_{43}, \tilde{A}_{43}, \tilde{B}_{39}) \tag{7.131}$$

$$\begin{aligned}
(B_{43} - C_{43})^{\ggg hs4} - (\tilde{B}_{43} - \tilde{C}_{43})^{\ggg hs4} &= B_{39} - \tilde{B}_{39} + \\
&\quad h(C_{43}, D_{43}, A_{43}) - h(\tilde{C}_{43}, \tilde{D}_{43}, \tilde{A}_{43}) \quad (7.132) \\
(A_{47} - B_{43})^{\ggg hs1} - (\tilde{A}_{47} - \tilde{B}_{43})^{\ggg hs1} &= A_{43} - \tilde{A}_{43} + \\
&\quad h(B_{43}, C_{43}, D_{43}) - h(\tilde{B}_{43}, \tilde{C}_{43}, \tilde{D}_{43}) \quad (7.133) \\
(D_{47} - A_{47})^{\ggg hs2} - (\tilde{D}_{47} - \tilde{A}_{47})^{\ggg hs2} &= D_{43} - \tilde{D}_{43} + \\
&\quad h(A_{47}, B_{43}, C_{43}) - h(\tilde{A}_{47}, \tilde{B}_{43}, \tilde{C}_{43}) \quad (7.134) \\
(C_{47} - D_{47})^{\ggg hs3} - (\tilde{C}_{47} - \tilde{D}_{47})^{\ggg hs3} &= C_{43} - \tilde{C}_{43} + \\
&\quad h(D_{47}, A_{47}, B_{43}) - h(\tilde{D}_{47}, \tilde{A}_{47}, \tilde{B}_{43}) \quad (7.135) \\
(B_{47} - C_{47})^{\ggg hs4} - (B_{47} - \tilde{C}_{47})^{\ggg hs4} &= B_{43} - \tilde{B}_{43} + \\
&\quad h(C_{47}, D_{47}, A_{47}) - h(\tilde{C}_{47}, \tilde{D}_{47}, \tilde{A}_{47}) \quad (7.136) \\
0 &= A_{47} - \tilde{A}_{47} + \\
&\quad i(B_{47}, C_{47}, D_{47}) - i(B_{47}, \tilde{C}_{47}, \tilde{D}_{47}) \quad (7.137) \\
0 &= D_{47} - \tilde{D}_{47} + \\
&\quad i(A_{51}, B_{47}, C_{47}) - i(A_{51}, B_{47}, \tilde{C}_{47}) \quad (7.138) \\
0 &= C_{47} - \tilde{C}_{47} + X_{14} - \tilde{X}_{14}. \quad (7.139)
\end{aligned}$$

Two stages are distinguished in this phase of the attack. The first deals with the differential properties of the set of equations and the second deals with the solution of the set of difference equations.

Differential Analysis

Before proceeding to find solutions to equations (7.124) to (7.139) the following observations are made. Dobbertin defines the differences for the chaining variables B_{39}, A_{43}, C_{43} and D_{43} as:

$$\begin{aligned}
\tilde{A}_{43} &= A_{43} - \varepsilon_1^2 \quad (7.140) \\
\tilde{B}_{39} &= B_{39} - \varepsilon_2^2 \quad (7.141) \\
\tilde{C}_{43} &= C_{43} - \varepsilon_3^2 \quad (7.142) \\
\tilde{D}_{43} &= D_{43} - \varepsilon_4^2 \quad (7.143) \\
& \quad (7.144)
\end{aligned}$$

where:

$$\begin{aligned} \varepsilon_1^2 &= \texttt{0x40004000} && (7.145)\\ \varepsilon_2^2 &= \texttt{0x80004000} && (7.146)\\ \varepsilon_3^2 &= \texttt{0xFFFBFE00} && (7.147)\\ \varepsilon_4^2 &= \texttt{0x40000200}. && (7.148) \end{aligned}$$

The values chosen for ε_1^2, ε_2^2, ε_3^2, and ε_4^2 may be obtained from a differential attack. The differential attack is applied to equations (7.132) to (7.139). By starting at equation (7.139) and working back to equation (7.132), the following differential relationships are observed:

$$\begin{aligned} C_{47} - \tilde{C}_{47} &= \tilde{X}_{14} - X_{14} && (7.149)\\ D_{47} - \tilde{D}_{47} &= i(A_{51}, B_{47}, \tilde{C}_{47}) - i(A_{51}, B_{47}, C_{47}) && (7.150)\\ A_{47} - \tilde{A}_{47} &= i(B_{47}, \tilde{C}_{47}, \tilde{D}_{47}) - i(B_{47}, C_{47}, D_{47}) && (7.151)\\ B_{43} - \tilde{B}_{43} &= (B_{47} - C_{47})^{\ggg hs4} - (B_{47} - \tilde{C}_{47})^{\ggg hs4} - \\ &\quad h(C_{47}, D_{47}, A_{47}) + h(\tilde{C}_{47}, \tilde{D}_{47}, \tilde{A}_{47}) && (7.152)\\ C_{43} - \tilde{C}_{43} &= (C_{47} - D_{47})^{\ggg hs3} - (\tilde{C}_{47} - \tilde{D}_{47})^{\ggg hs3} - \\ &\quad h(D_{47}, A_{47}, B_{43}) + h(\tilde{D}_{47}, \tilde{A}_{47}, \tilde{B}_{43}) && (7.153)\\ D_{43} - \tilde{D}_{43} &= (D_{47} - A_{47})^{\ggg hs2} - (\tilde{D}_{47} - \tilde{A}_{47})^{\ggg hs2} - \\ &\quad h(A_{47}, B_{43}, C_{43}) + h(\tilde{A}_{47}, \tilde{B}_{43}, \tilde{C}_{43}) && (7.154)\\ A_{43} - \tilde{A}_{43} &= (A_{47} - B_{43})^{\ggg hs1} - (\tilde{A}_{47} - \tilde{B}_{43})^{\ggg hs1} - \\ &\quad h(B_{43}, C_{43}, D_{43}) + h(\tilde{B}_{43}, \tilde{C}_{43}, \tilde{D}_{43}) && (7.155)\\ B_{39} - \tilde{B}_{39} &= (B_{43} - C_{43})^{\ggg hs4} - (\tilde{B}_{43} - \tilde{C}_{43})^{\ggg hs4} - \\ &\quad h(C_{43}, D_{43}, A_{43}) + h(\tilde{C}_{43}, \tilde{D}_{43}, \tilde{A}_{43}) && (7.156) \end{aligned}$$

From equation (7.149) it is observed that:

$$\begin{aligned} C_{47} - \tilde{C}_{47} &= \Delta \\ &= \texttt{0x00000200} && (7.157) \end{aligned}$$

By setting A_{51}, B_{47} and $\tilde{C}_{47}$ to zero and observing the difference obtained from equation (7.149), equation (7.150) yields:

$$D_{47} - \tilde{D}_{47} = \texttt{0x00000200}. \qquad (7.158)$$

For randomly chosen values for A_{51}, B_{47} and $\tilde{C}_{47}$ the relationship defined in equation (7.158) holds with a probability of 19%. Consider equation (7.151). By setting B_{47}, $\tilde{C}_{47}$ and $\tilde{D}_{47}$ to zero and by observing the differential values defined by equations (7.157) and (7.158) the following difference is obtained from equation (7.151).

$$A_{47} - \tilde{A}_{47} = \texttt{0x00000000}. \tag{7.159}$$

For randomly chosen values of B_{47}, $\tilde{C}_{47}$ and $\tilde{D}_{47}$ the relationship defined by equation (7.159) holds with a 19% probability. By setting $\tilde{A}_{47}$ to one and B_{47}, $\tilde{C}_{47}$ and $\tilde{D}_{47}$ to zero, we obtain the following differential from equation (7.152).[1]

$$B_{43} - \tilde{B}_{43} = \texttt{0xFFFC0000}. \tag{7.160}$$

For randomly chosen values of $\tilde{A}_{47}$, B_{47}, $\tilde{C}_{47}$ and $\tilde{D}_{47}$ the relationship in equation (7.160) holds with a probability of 33%. By setting $\tilde{C}_{47}$ and $\tilde{D}_{47}$ to zero and by setting $\tilde{A}_{47}$ and $\tilde{B}_{43}$ to -1, the following relationship is observed from equation (7.153).

$$C_{43} - \tilde{C}_{43} = \texttt{0xFFFBFE00}. \tag{7.161}$$

The relationship in equation (7.161) holds with a probability of 11% if the values for $\tilde{C}_{47}$, $\tilde{D}_{47}$, $\tilde{A}_{47}$ and $\tilde{B}_{43}$ are chosen at random and the previously determined differential values are observed. From equation (7.154) the following differential relationship is observed by setting $\tilde{D}_{47}$ equal to zero, $\tilde{A}_{47}$ and $\tilde{B}_{47}$ to -1 and $\tilde{C}_{43}$ to $-\texttt{0xFFFBFE00}$.

$$D_{43} - \tilde{D}_{43} = \texttt{0x40000200}. \tag{7.162}$$

This relationship holds with a probability of 8% for randomly chosen values for $\tilde{D}_{47}$, $\tilde{A}_{47}$, $\tilde{B}_{47}$ and $\tilde{C}_{43}$. If the following settings are made:

$$\begin{aligned}
\tilde{A}_{47} &= \texttt{0x00000000} \\
\tilde{B}_{43} &= \texttt{0x00000000} \\
\tilde{C}_{43} &= -\texttt{0xFFFBFE00} \\
\tilde{D}_{43} &= -(\texttt{0x40000200} + \texttt{0x40000200}))
\end{aligned}$$

[1]This relationship only holds when $(-C_{47})^{\ggg hs4} = -(C_{47})^{\ggg hs4}$

Equation (7.155) yields the following differential.

$$A_{43} - \tilde{A}_{43} = \texttt{0x40004000}. \tag{7.163}$$

Equation (7.163) holds with a probability of 5% if $\tilde{A}_{47}$, $\tilde{B}_{43}$, $\tilde{C}_{43}$ and $\tilde{D}_{43}$ are chosen at random and the previously computed differentials are used. Consider equation (7.156). By setting:

$$\begin{aligned}
\tilde{B}_{43} &= \texttt{0x00000000} \\
\tilde{C}_{43} &= -\texttt{0xFFFBFE00} \\
\tilde{D}_{43} &= -(\texttt{0x40000200} + \texttt{0x40000200}) \\
\tilde{A}_{43} &= \texttt{0x40000200} - \texttt{0x40004000}
\end{aligned}$$

equation (7.156) yields the following differential:

$$B_{39} - \tilde{B}_{39} = \texttt{0x80084000}. \tag{7.164}$$

This equation holds with a probability of 1.8% for randomly chosen values for $\tilde{B}_{43}$, $\tilde{C}_{43}$, $\tilde{D}_{43}$ and $\tilde{A}_{43}$.

Let $\Pr(\Delta_{i,j,k,l})$ denote the probability that a differential associated with a specific step holds. The differential attack is summarised in Table 7.1.

$\{i,j,k,l\}$	$A_i - \tilde{A}_i$	$B_j - \tilde{B}_j$	$C_k - \tilde{C}_k$	$D_l - \tilde{D}_l$	$\Pr(\Delta_{i,j,k,l})$
$\{51,47,51,51\}$	0	0	0	0	—
$\{51,47,47,51\}$	0	0	**0x00000200**	0	1.0
$\{51,47,47,47\}$	0	0	0x00000200	**0x00000200**	0.19
$\{47,47,47,47\}$	**0x00000000**	0	0x00000200	0x00000200	0.19
$\{47,43,47,47\}$	0x00000000	**0xFFFC0000**	0x00000200	0x00000200	0.33
$\{47,43,43,47\}$	0x00000000	0xFFFC0000	**0xFFFBFE00**	0x00000200	0.11
$\{47,43,43,43\}$	0x00000000	0xFFFC0000	0xFFFBFE00	**0x40000200**	0.08
$\{43,43,43,43\}$	**0x40004000**	0xFFFC0000	0xFFFBFE00	0x40000200	0.05
$\{43,39,43,43\}$	0x40004000	**0x80084000**	0xFFFBFE00	0x40000200	0.018

Table 7.1: Differential Attack: MD5

Let $\Pr(\Delta)$ denote the probability that the differential pattern in Table 7.1 holds. Assuming statistical independence between successive steps the probability that the differential attack holds is approximately:

$$\Pr(\Delta) = 1^{-7}.$$

However, a practical implementation has shown that the assumption of statistical independence is not valid. There exists a high probability that, given that one differential is satisfied, that the following differential will also be satisfied. A practical implementation of the differential attack has shown that the probability that the differential attack described in Table 7.1 holds is approximately:

$$\Pr(\Delta) = \frac{1}{26000}.$$

It appears that there exists a number of differentials which may be used instead of those shown in Table 7.1.

If the following constraints are imposed on B_{39}, A_{43}, C_{43} and D_{43} the probability that the differential attack holds is increased. Choose A_{43} at random except for bits 10, 15 and 31 which should be set to one and bit 19 which should be set to zero. C_{43} may be chosen at random except for bits 15 and 31 which should be set to one and bits 10 and 19 which should be set to zero[2]. Let D_{43} be specified by:

$$D_{43} = (A_{43} + (\varepsilon_4^2 - \varepsilon_1^2)^{\lll 31}) \oplus \gamma_1 \tag{7.165}$$

where γ_1 is a 32 bit binary vector with a low Hamming weight. In a similar fashion let B_{39} be defined as:

$$B_{39} = (\varepsilon_2^2 - 1)^{\lll 31}) \oplus \gamma_2 \tag{7.166}$$

where γ_2 denotes a 32 bit binary vector of low Hamming weight $(\gamma_1 \neq \gamma_2$.

The condition imposed by expression (7.69) for determining an internal collision may be re-written as:

$$0 = COMPRESS_{7.51}^{7.44} - \widetilde{COMPRESS}_{7.51}^{7.44} \tag{7.167}$$

[2]The bits in the 32 bit words are numbered from the LSB to the MSB

If the constraints described above are imposed on B_{39}, A_{43}, C_{43} and D_{43} the probability that expression 7.167 holds is approximately:

$$\Pr(\Delta) = \frac{1}{1500}.$$

Establishing an Inner Collision

Thus, the differential attack described here allows the problem of finding a solution to expression (7.69) to be reduced to finding a solution to the following expression:

$$COMPRESS_{7.44}^{7.36} - \widetilde{COMPRESS}_{7.44}^{7.36} = \{\varepsilon_1^2, \varepsilon_2^2, \varepsilon_3^2, \varepsilon_4^2\}. \qquad (7.168)$$

Thus the set of difference equations which has to be solved, may be simplified to:

$$\begin{aligned}
(B_{35} - C_{35})^{\ggg hs4} - (\tilde{B}_{35} - C_{35})^{\ggg hs4} &= X_{14} - \tilde{X}_{14} && (7.169)\\
(A_{39} - B_{35})^{\ggg hs1} - (\tilde{A}_{39} - \tilde{B}_{35})^{\ggg hs1} &= h(B_{35}, C_{35}, D_{35}) - h(\tilde{B}_{35}, C_{35}, D_{35}) && (7.170)\\
(D_{39} - A_{39})^{\ggg hs2} - (\tilde{D}_{39} - \tilde{A}_{39})^{\ggg hs2} &= h(A_{39}, B_{35}, C_{35}) - h(\tilde{A}_{39}, \tilde{B}_{35}, C_{35}) && (7.171)\\
(C_{39} - D_{39})^{\ggg hs3} - (\tilde{C}_{39} - \tilde{D}_{39})^{\ggg hs3} &= h(D_{39}, A_{39}, B_{35}) - h(\tilde{D}_{39}, \tilde{A}_{39}, \tilde{B}_{35}) && (7.172)\\
(B_{39} - C_{39})^{\ggg hs4} - (\tilde{B}_{39} - \tilde{C}_{39})^{\ggg hs4} &= B_{35} - \tilde{B}_{35} + \\
&\quad h(C_{39}, D_{39}, A_{39}) - h(\tilde{C}_{39}, \tilde{D}_{39}, \tilde{A}_{39}) && (7.173)\\
(A_{43} - B_{39})^{\ggg hs1} - (\tilde{A}_{43} - \tilde{B}_{39})^{\ggg hs1} &= A_{39} - \tilde{A}_{39} + \\
&\quad h(B_{39}, C_{39}, D_{39}) - h(\tilde{B}_{39}, \tilde{C}_{39}, \tilde{D}_{39}) && (7.174)\\
(D_{43} - A_{43})^{\ggg hs2} - (\tilde{D}_{43} - \tilde{A}_{43})^{\ggg hs2} &= D_{39} - \tilde{D}_{39} + \\
&\quad h(A_{43}, B_{39}, C_{39}) - h(\tilde{A}_{43}, \tilde{B}_{39}, \tilde{C}_{39}) && (7.175)\\
(C_{43} - D_{43})^{\ggg hs3} - (\tilde{C}_{43} - \tilde{D}_{43})^{\ggg hs3} &= C_{39} - \tilde{C}_{39} + \\
&\quad h(D_{43}, A_{43}, B_{39}) - h(\tilde{D}_{43}, \tilde{A}_{43}, \tilde{B}_{39}). && (7.176)
\end{aligned}$$

Additional simplifications to equations (7.124) – (7.139) can be achieved by making the following observations:

$$\begin{aligned}
(B_{35} - C_{35})^{\ggg hs4} - (\tilde{B}_{35} - C_{35})^{\ggg hs4} &= X_{14} - \tilde{X}_{14} && (7.177)\\
(B_{35} - C_{35})^{\ggg 23} - (\tilde{B}_{35} - C_{35})^{\ggg 23} &= -1^{\lll 9} && (7.178)\\
B_{35} &= ((\tilde{B}_{35} - C_{35})^{\ggg 23} - 1^{\lll 9})^{\lll 23} + C_{35}. && (7.179)
\end{aligned}$$

Equation (7.179) may, with high probability, be written as:

$$\begin{aligned} B_{35} &= ((\tilde{B}_{35} - C_{35})^{\lll 9} - 1^{\lll 9})^{\lll 23} + C_{35}. && (7.180)\\ B_{35} &= \tilde{B}_{35} - C_{35} - 1 + C_{35}. && (7.181)\\ B_{35} &= \tilde{B}_{35} - 1. && (7.182)\end{aligned}$$

By setting

$$B_{35} = -1. \tag{7.183}$$

and

$$\tilde{B}_{35} = 0. \tag{7.184}$$

equation (7.179) is satisfied. These choices inherently simplify equation (7.125) as follows.

$$\begin{aligned} (A_{39} - B_{35})^{\ggg hs1} - (\tilde{A}_{39} - \tilde{B}_{35})^{\ggg hs1} &= h(-1, C_{35}, D_{35}) - h(0, C_{35}, D_{35}) && (7.185)\\ (A_{39} - B_{35})^{\ggg hs1} - (\tilde{A}_{39} - \tilde{B}_{35})^{\ggg hs1} &= \overline{C_{35} \oplus D_{35}} - C_{35} \oplus D_{35} && (7.186)\end{aligned}$$

with the use of the following identity:

$$\begin{aligned} X \oplus Y + \overline{X \oplus Y} = 1 && (7.187)\\ \overline{X \oplus Y} = 1 - X \oplus Y && (7.188)\end{aligned}$$

equation (7.186) reduces to:

$$\begin{aligned} 1 - (A_{39} - B_{35})^{\ggg hs1} + (\tilde{A}_{39} - \tilde{B}_{35})^{\ggg hs1} &= 2 \cdot (C_{35} \oplus D_{35}) && (7.189)\\ C_{35} \oplus D_{35} &= (1 - (A_{39} - B_{35})^{\ggg hs1} - \\ &\quad (\tilde{A}_{39} - \tilde{B}_{35})^{\ggg hs1})^{\ggg 1} && (7.190)\end{aligned}$$

thus D_{35} may be obtained from:[3]

$$D_{35} = (1 - (A_{39} - B_{35})^{\ggg hs1} - (\tilde{A}_{39} - \tilde{B}_{35})^{\ggg hs1})^{\ggg 1} \oplus C_{35}. \tag{7.191}$$

[3]Note that equation 7.191 only holds if the right hand side of equation (7.189) is even.

Now consider equation (7.172). Remember that:

$$h(A, B, C) = A \oplus B \oplus C.$$

Thus equation (7.172) may be written as:

$$(C_{39} - D_{39})^{\ggg hs3} - (\tilde{C}_{39} - \tilde{D}_{39})^{\ggg hs3} = D_{39} \oplus A_{39} \oplus B_{35} - \tilde{D}_{39} \oplus \tilde{A}_{39} \oplus \tilde{B}_{35} \quad (7.192)$$

Let:

$$\begin{aligned} X &= D_{39} \oplus A_{39} \oplus B_{35} & (7.193) \\ \tilde{X} &= \tilde{D}_{39} \oplus \tilde{A}_{39} \oplus \tilde{B}_{35} & (7.194) \end{aligned}$$

From equations (7.183) and (7.184) equations (7.193) and (7.194) may be simplified to:

$$\begin{aligned} X &= \overline{D_{39} \oplus A_{39}} & (7.195) \\ \tilde{X} &= \tilde{D}_{39} \oplus \tilde{A}_{39} & (7.196) \end{aligned}$$

Equation 7.192 may now be written as:

$$X - \tilde{X} = (C_{39} - D_{39})^{\ggg hs3} - (\tilde{C}_{39} - \tilde{D}_{39})^{\ggg hs3}. \quad (7.197)$$

Equation 7.197 may be written, with a probability of approximately 25% as:

$$X - \tilde{X} = ((C_{39} - \tilde{C}_{39}) - (D_{39} - \tilde{D}_{39}))^{\ggg hs3}. \quad (7.198)$$

Equation (7.173) may now be written as:

$$\begin{aligned} (B_{39} - C_{39})^{\ggg hs4} - (\tilde{B}_{39} - \tilde{C}_{39})^{\ggg hs4} &= B_{35} - \tilde{B}_{35} + \\ &\quad h(C_{39}, \overline{X}, 0) - h(\tilde{C}_{39}, \tilde{X}, 0) \quad (7.199) \end{aligned}$$

Equation (7.174) may now be written as:

$$\begin{aligned} (A_{43} - B_{39})^{\ggg hs1} - (\tilde{A}_{43} - \tilde{B}_{39})^{\ggg hs1} &= (\overline{X} \oplus D_{39} - \tilde{X} \oplus \tilde{D}_{39}) + h(B_{39}, C_{39}, D_{39}) \\ &\quad - h(\tilde{B}_{39}, \tilde{C}_{39}, \tilde{D}_{39}) \quad (7.200) \end{aligned}$$

Thus:

$$A_{39} = \overline{X} \oplus D_{39} \quad (7.201)$$

$$\tilde{A}_{39} = \tilde{X} \oplus \tilde{D}_{39} \quad (7.202)$$

Dobbertin proposes that Algorithm 7.3 is used to determine a solution to expression (7.168).

Algorithm 7.3 *Construction of Inner Collision: Phase II*

1. *Choose B_{35} and $\tilde{B}_{35}$ as specified by equations (7.183) and (7.184)*
2. *Find values for A_{43}, B_{39}, C_{43}, D_{43} which will satisfy the differential attack summarised in Table 7.1.*
3. *Determine $C_{39} - \tilde{C}_{39}$ from equation (7.176).*
4. *Choose a random value for C_{39} and determine $\tilde{C}_{39}$ from the result obtained in step 3. Now determine $D_{39} - \tilde{D}_{39}$ from equation (7.176).*
5. *With $D_{39} - \tilde{D}_{39}$ in hand determine all possible solutions to C_{39} and $\tilde{C}_{39}$ using an iterative search procedure.*
6. *Calculate $X - \tilde{X}$ given that the assumptions in (7.198) holds.*
7. *From equation (7.199) determine all valid solutions to X and $\tilde{X}$.*
8. *Determine solutions for D_{39} and $\tilde{D}_{39}$ from equation (7.200) using an iterative approach.*
9. *A_{39} and $\tilde{A}_{39}$ may be calculated from equations (7.201) and (7.202).*
10. *Determine if the assumption in equation (7.198) holds.*
    (a) *If the assumption in expression (7.198) does not hold return to step 1.*
    (b) *If the assumption in (7.198) holds, determine all possible solutions to C_{35} using equation (7.171).*
11. *If solutions for C_{35} exist determine D_{35} from (7.191).*
12. *If a valid solution D_{35} is found an inner collision for the second round was found.*

Dobbertin's implementation of this attack is attached as Appendix E.2.

7.5.5 Phase III: Establishing a Connection

The third phase requires that the solutions to the sets of equations obtained from the previous two phases are connected. When commencing with the third phase of the attack, the chaining variables C_{15}, B_{15}, A_{19}, D_{19}, C_{19} B_{19} and A_{23} are known from phase one of the attack. The chaining variables C_{35}, D_{35}, A_{39}, B_{39}, C_{39}, D_{39}, A_{43} and D_{43} are known from phase two of the attack. A number of message words and chaining variables may now be computed. Message words X_1, X_6, X_{11}, X_0 and X_5 may be computed from equations (7.17), (7.18), (7.19), (7.20) and (7.21). Likewise message words X_4, X_7, X_{10} and X_{13} may be obtained from equations (7.38), (7.39), (7.40) and (7.41). In addition chaining variables D_{23} and D_{43} are be obtained from equations (7.22) and (7.41) respectively.

A connection is obtained if solutions to X_2, X_3, X_8 and X_{12} is found such that equations (7.203)–(7.203) holds.

$$\begin{aligned}
C_{27} &= D_{27} + (C_{23} + g(D_{27}, A_{27}, B_{23}) + X_3 + T_{26})^{\lll gs3} && (7.203)\\
B_{27} &= C_{27} + (B_{23} + g(C_{27}, D_{27}, A_{27}) + X_8 + T_{27})^{\lll gs4} && (7.204)\\
A_{31} &= B_{27} + (A_{27} + g(B_{27}, C_{27}, D_{27}) + X_{13} + T_{28})^{\lll gs1} && (7.205)\\
D_{31} &= A_{31} + (D_{27} + g(A_{31}, B_{27}, C_{27}) + X_2 + T_{29})^{\lll gs2} && (7.206)\\
C_{31} &= D_{31} + (C_{27} + g(D_{31}, A_{31}, B_{27}) + X_7 + T_{30})^{\lll gs3} && (7.207)\\
B_{31} &= C_{31} + (B_{27} + g(C_{31}, D_{31}, A_{31}) + X_{12} + T_{31})^{\lll gs4} && (7.208)\\
A_{35} &= B_{31} + (A_{31} + h(B_{31}, C_{31}, D_{31}) + X_5 + T_{32})^{\lll hs1} && (7.209)\\
D_{35} &= A_{35} + (D_{31} + h(A_{35}, B_{31}, C_{31}) + X_8 + T_{33})^{\lll hs2} && (7.210)\\
C_{35} &= D_{35} + (C_{31} + h(D_{35}, A_{35}, B_{31}) + X_{11} + T_{34})^{\lll hs3} && (7.211)
\end{aligned}$$

Equations (7.203)–(7.203) are bounded by the chaining variables A_{27}, B_{23}, C_{23} and D_{23} as obtained from phase one of the attack, as well as by chaining variables A_{35}, B_{31}, C_{35} and D_{35}, obtained from phase two of the attack. Thus the only chaining variables which may be manipulated without affecting the previous phases of the attack, are A_{31}, B_{27}, C_{27} and D_{31}. An additional constraint is imposed by the fact that X_5, X_7, X_{11} and X_{13} are determined by phase one and two of the attack. This leaves only four message words namely X_2, X_3, X_8 and X_{12}, which may be used to establish a connection. The fact that expression (7.68) is simplified to expression (7.111) gives an attacker additional degrees of freedom and allows, to a limited extent, the manipulation of chaining variables B_{23} C_{23} and D_{27}. Note that D_{23}, which is associated with the first stage of the attack, depends on X_{10} which is obtained from the second phase of the attack. Thus the chaining variables in the two previous phases are interdependent on each other. This requires that the results obtained from the previous phases has to be manipulated simultaneously when attempting to establish a connection between the two phases.

Three stages are identified in the third phase of the attack. The first stage is concerned with finding suitable values for the chaining variables C_{23}, B_{23}, A_{27} and D_{27}. The second stage requires that a connection be made between phase one and two. During the third stage it is required that the existence of an inner collision obtained for the second phase is verified.

Stage 1

Remember that the attack on the first phase is simplified in Section 7.5.3 by requiring that condition (7.111) instead of condition (7.68) has to be met. From Section 7.5.3 it is known that if condition (7.111) is met, condition (7.68) holds with a high probability. The following procedure is proposed by Dobbertin to find suitable values for C_{23}, B_{23}, A_{27} and D_{27}.

Algorithm 7.4 *Stage 1: Determine if inner collision for Phase I*

1. *Determine D_{23} from equation (7.22).*
2. *Choose X_{15} at random.*
3. *Determine C_{23} and B_{23} from equations (7.23) and (7.24).*
4. *Set A_{23} equal to B_{23}.*
5. *Calculate X_9 from equation (7.25)*
6. *Set D_{27} =* 0xFFFFFFFF.
7. *Calculate X_{14} from (7.26).*
8. *Determine if expression (7.68) holds.*
    (a) *If condition (7.68) does not hold, restart from step 1.*
    (b) *If condition (7.68) does hold, proceed to stage two of phase three.*

The choice for D_{27} ensures that the following relationship holds:

$$B_{27} = g(B_{27}, C_{27}, D_{27}). \qquad (7.212)$$

Stage 2

The second stage is directly concerned with establishing a connection between the two phases. The following simplification to equation (7.209) is of particular importance:

$$\begin{aligned}
(A_{35} - B_{31})^{\ggg hs1} - A_{31} - X_5 - T_{32} &= h(B_{31}, C_{31}, D_{31}) && (7.213)\\
(A_{35} - B_{31})^{\ggg hs1} - A_{31} - X_5 - T_{32} &= B_{31} \oplus C_{31} \oplus D_{31} && (7.214)\\
D_{31} &= ((A_{35} - B_{31})^{\ggg hs1} - A_{31} - X_5 - T_{32}) &&\\
&\quad \oplus B_{31} \oplus C_{31}. && (7.215)
\end{aligned}$$

Dobbertin proposes the following algorithm to establish a connection between the two phases.

Algorithm 7.5 *Second Stage: Establish Connection*

1. *Set* $B_{31} = B_{35} - X_{14}$.
2. *Determine* C_{31} *from equation (7.211).*
3. *Choose a random value for* $B_{27_{Basic}}$. *Choose* B_{27} *with a Hamming distance of 1 from* $B_{27_{Basic}}$.
4. *Calculate* A_{31} *from equation (7.205).*
5. *Calculate* D_{31} *from equation (7.215).*
6. *Determine* C_{27} *from equation (7.207).*
7. *Calculate* X_8^1 *from equation (7.204) and* X_8^2 *from equation (7.210).*
   (a) *If equations (7.204) and (7.210) yields the same result for* X_8 *a connection was found.*

   (b) *If equations (7.204) and (7.210) yields different results for X_8 continue from step 8.*

8. *Determine the Hamming distance between the two values calculated for X_8.*

   (a) *If the current Hamming distance is less than the previous Hamming distance, save the current Hamming distance and set $B_{27_{Basic}} = B_{27}$. Return to step 4.*

   (b) *If the current Hamming distance is not less than the previous Hamming distance, proceed from step 9.*

9. *Set: $C_{23} = C_{23} - (X_8^1 - X_8^2)$, $B_{23} = B_{23} - (X_8^1 - X_8^2)$ and recompute D_{27}. Confirm if equation (7.212) holds.*

   (a) *If equation (7.212) does not hold, restart from step 1.*

   (b) *If equation (7.212) does hold, proceed to step 10.*

10. *Determine if the changes made in step 9 still allows an inner collision for the first phase.*

    (a) *If not, restart from Algorithm 7.4.*

    (b) *If an internal collision is found, restart from step 4.*

Step 8 in Algorithm 7.5 is reminiscent of the continuous approximation used in phase one of the attack on MD5.

Upon successful completion of Algorithm 7.5 X_2, X_3, X_8 and X_{12} may be computed from equations (7.206), (7.203), (7.204) and (7.208) respectively. Once the first two stages have been completed, only the third stage of the third phase remains.

Stage 3

The third stage requires that the existence of an inner collision for the first two stages be confirmed. If an inner collision is not found, the attacker should restart from stage 1 of the third phase. If the inner collision for the second phase exists, a collision for the round function of MD5 was found. The message words and initial value for which the collision holds may now be calculated.

Example

The following collision was constructed for the round function of MD5 The initial value for which a collision was found is given by:

$$A0 = \texttt{0xF7987AA4}$$
$$B0 = \texttt{0x6EF00D2B}$$
$$C0 = \texttt{0xCAFBC0A2}$$
$$D0 = \texttt{0x7678589B}$$

The message is defined by:

$$X_0 = \texttt{0xAA1DDA5E} \qquad X_8 = \texttt{0x4C1E82F6}$$
$$X_1 = \texttt{0xD97ABFF5} \qquad X_9 = \texttt{0xAF5B46C9}$$
$$X_2 = \texttt{0xA5CA745B} \qquad X_{10} = \texttt{0x236BB992}$$
$$X_3 = \texttt{0xC9DDCECB} \qquad X_{11} = \texttt{0x6B7A669B}$$
$$X_4 = \texttt{0x1006363E} \qquad X_{12} = \texttt{0x40CC7121}$$
$$X_5 = \texttt{0x7218209D} \qquad X_{13} = \texttt{0xD93E0972}$$
$$X_6 = \texttt{0xE01C135D} \qquad X_{14} = \texttt{0x95FACCCD}$$
$$X_7 = \texttt{0x9DA64D0E} \qquad X_{15} = \texttt{0x72409780}$$

with

$$\tilde{X}_{14} = X_{14} + 1^{\lll 9}.$$

The common hash value obtained for these two messages is:

$$\text{Common Digest} = \texttt{0x88963A3F 0x4B40C08A 0xF529663D 0x0B410DAD}.$$

Note that due to the overlap in message words (specifically X_{10}) and the consequent interdependence of the chaining variables in phases one and two of the attack, considerable effort is required to find solutions to both phase one and two which will allow a successful connection to be made. Given a solution to the second phase the probability that the solution to the first phase will still result in an inner collision for the first phase is estimated at 25%.

The techniques used to construct a connection described in this section, manipulates the results obtained from phases one and two of the attack simultaneously. However the potential effort required to solve the sets of equations simultaneously is avoided by finding a solution to expression (7.111) instead of (7.68). This allows the limited manipulation of the chaining variables B_{23}, C_{23} and A_{27}. Thus, the chaining variables which are most influenced by the interdependence between phase one and two, may be manipulated to a limited extent. An implementation of the third phase of the attack is attached as Appendix E.3.

7.5.6 Determining if Solutions Exist

In Sections 7.5.3 and 7.5.4 it is assumed that an algorithm exists for determining if solutions to certain Boolean expressions exist. This section gives a description of such an algorithm as used by Dobbertin in the analysis of MD5. Dobbertin's attack requires the solution of sets of difference equations. From the previous sections it is observed that these equations are of the form:

$$T = f(a_1, b_1, x_1) - f(a_2, b_2, x_2) \tag{7.216}$$

or:

$$0 = f(a_1, b_1, x_1) - f(a_2, b_2, x_2) - T \tag{7.217}$$

with $f()$ a bitwise Boolean function which operates on binary words of length l. The variables x_1 and x_2 are related by:

$$x_2 = x_1 + \delta x$$

thus equation (7.217) may be written as:

$$0 = f(a_1, b_1, x_1) - f(a_2, b_2, x_1 + \delta x) - T \tag{7.218}$$

where a_1, b_1, a_2, b_2, T and δx are constants, x_1 is the unknown variable.

It is important to note that due to the nature of bitwise calculations, the i'th bit of the solution corresponds to the i'th bit of x_1. To be more specific, if a bit in position i is changed in x_1, this can at most cause bit changes ranging from bit position i up to the MSB (allowing for carry) when equation (7.218) is evaluated.

In [55] it is stated that all solutions of x_1 has the structure of a binary tree and can be computed using a bitwise recursive process. This process is presented as Algorithm 7.6.

Algorithm 7.6 *Recursive Search Procedure*

1. *Initialise a_1, b_1, a_2, b_2, T and x_1. Set δx to represent the desired difference between x_1 and x_2.*
2. *Set the current depth of the binary tree, i, to 0.*
3. *Determine the current depth of the binary tree.*
4. *If the current depth of the tree, i, equals l (the length of a binary word), record x_1 as a valid solution. Decrement the depth counter i by one. Return to the instruction (step) following the most recent entry or re-entry into Algorithm 7.6. If control is returned to the instruction following the original entry into Algorithm 7.6, Algorithm 7.6 is terminated.*
5. *Calculate the right hand side of equation (7.218). Determine if the i'th bit of the result equals zero.*
6. *If the i'th bit of the right hand side of equation (7.218) equals zero, increment the tree-depth counter i and re-enter Algorithm 7.6 at step 3.*
7. *Toggle the i'th bit in x_1. Re-compute the right hand side of equation (7.218). Determine if the i'th bit of the result equals zero.*
8. *If the i'th bit of the right hand side of equation (7.218) equals zero, increase the tree-depth counter i by one. Re-enter Algorithm 7.6 at step 3.*
9. *Toggle the i'th bit in x_1 and return the number of valid solutions found for x_1. Decrement the tree-depth counter i by one. Return to the instruction (step) following the most recent entry, or re-entry into Algorithm 7.6. If control is returned to the instruction following the original entry into Algorithm 7.6, Algorithm 7.6 is terminated.*

Algorithm 7.6 may be modified to search only for a limited number of solutions. In order to illustrate the relationship between Algorithm 7.6 and a binary tree, the following example is presented.

Example

Let l equal 4. Thus all the variables in equation (7.218) are 4 bit words. For these 4 bit words, let the LSB be numbered 0 and the MSB 3. Let $f()$ be a bitwise Boolean function defined by:

$$f(a, b, x) = (a \wedge x) \vee (\neg x \wedge b)$$

Let the subscript 2 denote the base 2. The following is an example of the application of Algorithm 7.6 to the four bit problem. Set:

$$\begin{aligned} T &= 0000_2 & a_2 &= a_1 \\ a_1 &= 0110_2 & b_2 &= b_1 \\ b_1 &= 1100_2 & \delta x &= 0101_2. \end{aligned}$$

Choose a random value for x_1, say 1001_2. A numerical example of the use of Algorithm 7.6 to find solutions to the given expressions is presented in Table 7.2.

Eight distinct headings are included in Table 7.2. The *Operation nr.* indicates the number of the operation. The *Step nr.* indicates which step in Algorithm 7.6 is associated with the given operation number. The column denoted i contains the tree-depth counter. The column marked $i = 4?$ indicates a test condition. The fifth column is marked x_1 and contains the present value of x_1. The column marked *Equation (7.218)* contains the value to which equation (7.218) evaluates. The *Resume From* column contains an operation number. Associated with the operation number is a step number. If the *Resume From* column contains an operation number, the next step which should be executed is the step following the step number associated with operation number in the *Resume From* column. The *Node* column contains the current node in the binary tree depicted in Figure 7.3. The variables which are updated in each operation are marked in gray.

Operation nr.	Step nr.	i	$i = 4$?	x_1	Equation (7.218)	Resume From Operation nr.	Node
1	2	0	No	1001_2	—	—	r
2	3	0	No	1001_2	—	—	r
3	5	0	No	1001_2	1110_2	—	r
4	6	1	No	1001_2	1110_2	—	B
5	3	1	No	1001_2	1110_2	—	B
6	5	1	No	1001_2	1110_2	—	B
7	7	1	No	1011_2	1010_2	—	B
8	9	0	No	1001_2	—	4	r
9	7	0	No	1000_2	0000_2	—	r
10	8	1	No	1000_2	0000_2	—	A
11	3	1	No	1000_2	0000_2	—	A
12	5	1	No	1000_2	0000_2	—	A
13	6	2	No	1000_2	0000_2	—	C
14	3	2	No	1000_2	0000_2	—	C
15	5	2	No	1000_2	0000_2	—	C
16	6	3	No	1000_2	0000_2	—	G
17	3	3	No	1000_2	0000_2	—	G
18	5	3	No	1000_2	0000_2	—	G
19	6	4	No	1000_2	0000_2	—	O
20	3	4	Yes	1000_2	0000_2	—	O
21	4	3	No	1000_2	0000_2	19	G
22	7	3	No	0000_2	0000_2	—	G
23	8	4	No	0000_2	0000_2	—	P
24	3	4	Yes	0000_2	0000_2	—	P
25	4	3	No	0000_2	0000_2	23	G
26	9	2	No	1000_2	—	16	C
27	7	2	No	1100_2	1000_2	—	C
28	8	3	No	1100_2	1000_2	—	H
29	3	3	No	1100_2	1000_2	—	H
30	5	3	No	1100_2	1000_2	—	H
31	7	3	No	0100_2	1000_2	—	H

Operation nr.	Step nr.	i	$i = 4?$	x_1	Equation (7.218)	Resume From Operation nr.	Node
32	9	2	No	1100_2	—	28	C
33	9	1	No	1000_2	—	13	A
34	7	1	No	1010_2	0000_2	—	A
35	8	2	No	1010_2	0000_2	—	D
36	3	2	No	1010_2	0000_2	—	D
37	5	2	No	1010_2	0000_2	—	D
38	6	3	No	1010_2	0000_2	—	I
39	3	3	No	1010_2	0000_2	—	I
40	5	3	No	1010_2	0000_2	—	I
41	6	4	No	1010_2	0000_2	—	T
42	3	4	Yes	1010_2	0000_2	—	T
43	4	3	No	1010_2	0000_2	41	I
44	7	3	No	0010_2	0000_2	—	I
45	8	4	No	0010_2	0000_2	—	S
46	3	4	Yes	0010_2	0000_2	—	S
47	4	3	No	0010_2	0000_2	45	I
48	9	2	No	1010_2	—	38	D
49	7	2	No	1110_2	1000_2	—	D
50	8	3	No	1110_2	1000_2	—	J
51	3	3	No	1110_2	1000_2	—	J
52	5	3	No	1110_2	1000_2	—	J
53	7	3	No	0110_2	1000_2	—	J
54	9	2	No	1110_2	—	50	D
55	9	1	No	1010_2	—	35	A
56	9	0	No	1000_2	—	10	r
57	9	0	No	1001_2	—	Exit	r

Table 7.2: Numerical Example: Recursive Search Procedure

An implementation of Algorithm 7.6 with reference to the example given above, is listed in Appendix E.

All possible values for x_1 may be represented as a binary tree (see Figure 7.3).

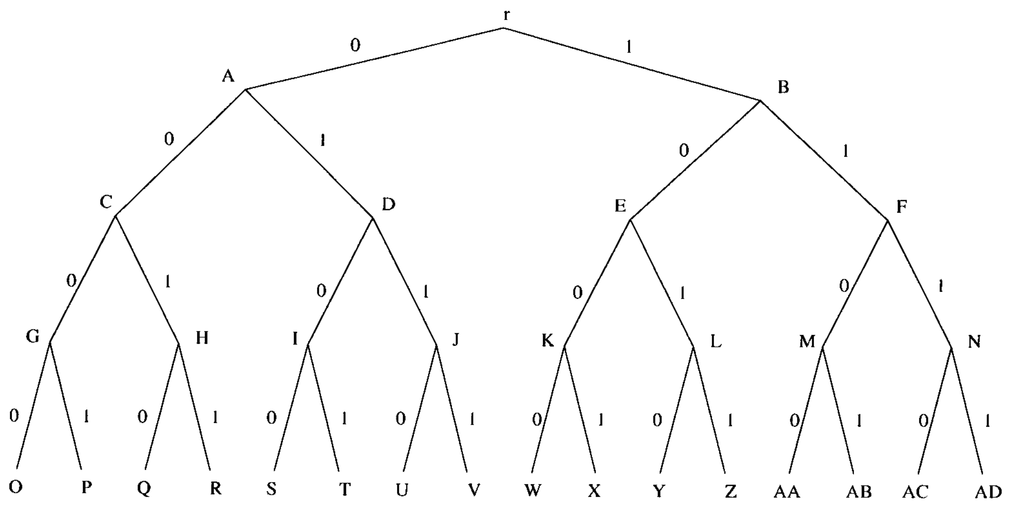


Figure 7.3: Binary Tree Representation of Four Bit Variable

Since all possible values of x_1 are included in this tree, all possible solutions to equation (7.218) are also included in this tree. The relationship between the binary tree in Figure 7.3 and Algorithm 7.6 may be observed from the above example by tracing the node positions at each step of the example. It is noted that if the children of a node (for instance node B) does not yield solutions, the entire branch may be pruned from the search space. Thus Algorithm 7.6 in effect searches through a binary tree. The pruned binary tree which contains only the solutions to the above example is shown in Figure 7.4.

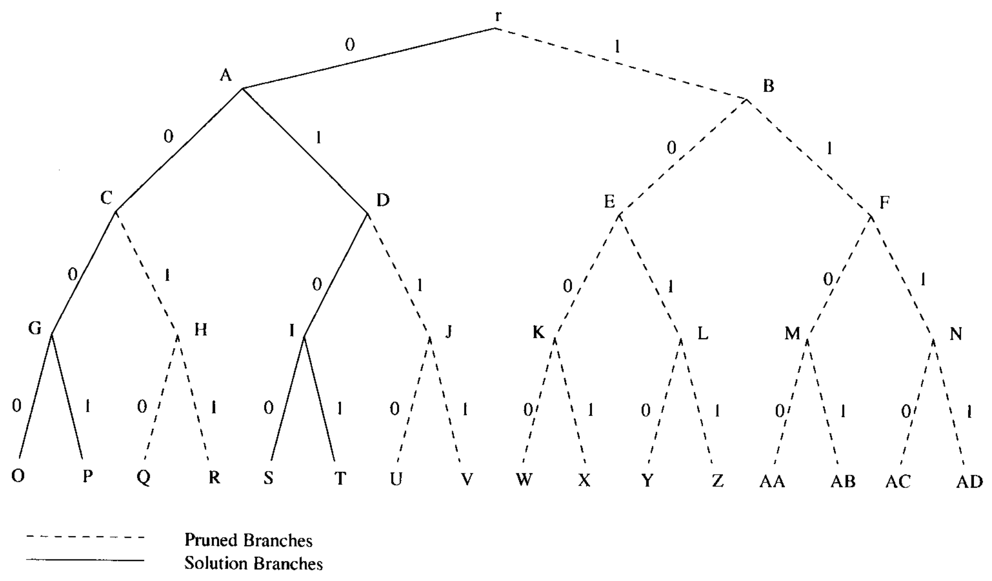


Figure 7.4: Pruned Binary Tree with Solutions

It should be remembered that Algorithm 7.6 only yields results if the Boolean functions operate bitwise on a multiple bit word. Any rotations or other sources of diffusion in the Boolean function, renders the above approach ineffective. The power of this search algorithm lies in the ability to reject entire branches at a time (e.g. in operation nr. 3, all the children nodes which are attached to node A are immediately rejected as possible solutions). The ability to reject entire subsets of possible solutions has the result that the existence of solutions, as well as the solutions themselves, can be determined quickly and efficiently.

7.5.7 Conclusion

This chapter contains a description of the cryptanalysis of MD5 as developed by Dobbertin. The attack on MD5 is similar to the attack on MD4 in a number of aspects. Both attacks require that sets of difference equations should be solved. Furthermore both attacks rely, to some extent, upon the fact that certain differential patterns are more likely to propagate than others. Furthermore both attacks depend on the fact that the Boolean expressions used in the compress functions may be manipulated. In both attacks, the fact that the rotation operations may be separated from the Boolean equations by the use of counter rotations allows the attacker to establish and solve equations of a certain form.

The attack on MD5 is however considerably more complex than the attack on MD4. The additional complexity of this attack is reflected in the amount of time required to find a collision for the compress function of MD5 when it is compared to the time required to find a collision for the compress function of MD4. The additional complexity appears to be due mainly to the use of an additional round. A further contribution to the complexity is the use of the XOR function in the third round. In the attack on MD4, Dobbertin avoids dealing with the XOR function (except briefly during the differential attack on MD4 [14]). In the attack on MD5 this is not the case. The contribution of the constants to the security of MD5 is considered to be low. The constants are only brought into play during the third phase of the attack, when a connection between the results obtained for phase one and two are established. Certain choices and conditions are imposed which appears to speed up the process of finding an inner collision for the second phase of the attack. It is unclear why these choices are made.

Of particular interest is the algorithm used by Dobbertin to determine if solutions to Boolean equations of a specific form exist. This algorithm is presented in Section 7.5.6. It appears that this algorithm is particularly suited to evaluate Boolean equations which has no diffusion properties over a certain number of bits. It is therefore particularly useful when dealing with bitwise operations as used in MD4 and MD5. If other cryptographic primitives, which utilises bitwise operations are analysed, this algorithm may be a useful tool. It is not known if this algorithm is described in the literature.

7.6 ACKNOWLEDGMENTS

I would like to make use of this opportunity to thank Antoon Bosselaers and the personnel at the COSIC group at the Katholieke Universiteit Leuven for their assistance and support during this investigation into the cryptanalysis of MD4 and MD5. The algorithms and techniques presented in this chapter were derived from the programs written by Dobbertin and the notes of Bosselaers.

CHAPTER 8: GENERALISED ANALYSIS OF THE MD4 FAMILY OF DEDICATED HASH FUNCTIONS

8.1 INTRODUCTION

The vast majority of hash functions (MDCs and MACs) are based on the iterated model known as the Dåmgard-Merkle scheme [22], [23]. This generalised construction is discussed in Chapter 5 Section 5.3. It was proved by Dåmgard that the security of the overall construction relies on the security of the compress function. Dedicated hash functions, including the MDx family of hash functions, are primarily based on the iterative model presented by Dåmgard and Merkle.

In 1996 Dobbertin presented an attack against RIPEMD-128 and MD4 [54], [14]. These attacks showed that it is possible to construct collisions for MD4 and RIPEMD-128. In addition to the above attacks Dobbertin presented an attack on the compress function of MD5. This attack demonstrated that it is computationally feasible to establish collisions for the compress function of MD5 [12], [55].

The similarity in the structure of the hash functions (MD4, MD5 and RIPEMD-128) suggests a common factor in the attacks. If a common factor could be found in all of the attacks, it may be indicative of a weakness in the design of an entire hash function family. Once this weakness is identified it may be possible to derive design criteria for dedicated hash functions. Hash functions designed according to these design criteria would be immune to these attacks.

For these reasons the attacks against MD4 and MD5 were reconstructed and studied (Chapter 6 and 7). Many interesting techniques and properties were discovered in the course of this analysis. Specifically a technique was discovered to speed-up the construction of collisions against MD4. This work resulted in a publication [56].

In this chapter the attacks on MD4 and MD5 are generalised to provide a framework for the analysis of any iterated cryptographic hash function.

8.2 GENERALISED ATTACKS

The attacks described by Dobbertin applies directly to the compress function of the hash functions. It is the aim of these attacks to find two messages M and $\tilde{M}$ with length equal to a single block such that:

$$f(IV, M) = f(IV, \tilde{M})$$

with $f()$ the compress function and IV the initial value used. The compress functions of the dedicated hash function under consideration are constructed by applying a number of steps iteratively. Each step may be expressed as an equation containing Boolean mappings, rotation operators, additive constants and addition $\bmod\ 2^{32}$ operators.

Let $f_i^j()$ represents the application of the compress function from step i to step j. Before proceeding it is appropriate to state the following definitions.

Definition 1 (Inner-Collision) *An inner-collision is defined if, between steps i and j of the compress function, $f_i^j(C, M) = f_i^j(C, \tilde{M})$. C_i is the internal chaining variable and is given by $C_i = f_0^i(IV, M) = f_0^i(IV, \tilde{M})$.*

Definition 2 (Almost Inner-Collision) *An almost inner-collision is defined if, between steps i and j, of the compress function $f_i^j(C, M) = f_i^j(C, \tilde{M}) + \Delta_i$ with Δ_i a specified difference. As before C_i is the internal chaining variable.*

The attacks by Dobbertin exploit the fact that

$$f(IV, M) = f(IV, \tilde{M})$$

holds if

$$\begin{array}{rcl}
f_0^i(IV, M) & = & f_0^i(IV, \tilde{M}) \\
f_{i+1}^j(C_i, M) & = & f_{i+1}^j(C_i, \tilde{M}) \\
f_{j+1}^k(C_j, M) & = & f_{j+1}^k(C_j, \tilde{M}) \\
\vdots & \vdots & \vdots \\
f_{k+1}^l(C_k, M) & = & f_{k+1}^l(C_k, \tilde{M})
\end{array}$$

for a compression function defined by l steps. Thus a number of consecutive inner-collisions may be used to construct a collision. It is noted that an inner-collision may be constructed from an almost inner-collision if differences and chaining variables are found such that:

$$\begin{aligned} f_i^j(C_i, M) &= f_i^j(C_i, \tilde{M}) + \Delta_j \\ f_j^k(C_j + \Delta_j, M) &= f_j^k(C_j, \tilde{M}) \end{aligned}$$

The attacks defined by Dobbertin are focused on finding inner-collisions and almost inner-collisions for sections of the compress function which would result in the construction of collisions for the entire compress function. These attacks share two elements. The first element deals with the derivation of sets of difference equations. The second element requires a solution to these sets of difference equations.

8.2.1 Difference Equations

The chaining variables in MD4 and MD5 are 32 bit words. For these hash functions a difference equation is defined as the difference mod 2^{32} between two expressions. This may be written as:

$$\Delta E = \left[E_1(M_i) - E_2(\tilde{M}_i)\right] \bmod 2^{32}. \qquad (8.1)$$

The first expression, $E_1(M_i)$, describes a given step. The variables in this expression are associated with the first message. The second expression, $E_2(\tilde{M}_i)$, describes the same step as the first. The variables used in the second expression are associated with the second message. Expression 8.1 is referred to as a difference equation. A set of difference equations is obtained if difference equations can be derived for a number of consecutive steps in the hash function.

Obtaining a set of difference equations for all the steps of a hash function is possible but not recommended since the dedicated hash functions described in the literature has upwards of 48 steps (MD4), and writing difference equations for all the steps results in a large number of interrelated non-linear equations which are difficult to solve.

For this reason the difference between two messages is restricted to a single message word. In MD4, MD5 message words are re-used in consecutive rounds. The order in which the message words are processed is changed for each round. In the analysis of MD4 and MD5 the message words in which the differences occur are selected according to the number of steps separating the occurrence of the given word in consecutive rounds.

There exist no exact rule for selecting the message word which will be altered. Instead a general guideline is that the number of steps between the occurrence of a given word should not be too small (less than four steps for MD4 and MD5) since this makes it difficult to derive a set of solvable difference equations. At the same time the number of steps between the occurrence of two steps should not be too large since this increases the number of variables which has to be solved.

In hash functions with an even number of rounds, a trade-off between the number of steps separating the message word in each pair of rounds has to be found. The difference equations should be setup to yield inner-collisions for each pair of rounds.

If the number of rounds are uneven, difference equations should be established for any combination of even rounds. One of the sets of difference equations should be setup to result in an almost inner-collision. This almost inner-collision should be specified in such a manner that a difference pattern propagates and interacts with the message word in the unmatched round. This interaction should result in a collision or inner-collision. As an example consider MD4 where it was shown that

$$f(IV, M) = f(IV, \tilde{M})$$

holds if:

$$\begin{aligned} f_{12}^{19}(C_{12}, M) &= f_{12}^{19}(C_{12}, \tilde{M}) + \Delta_{19} \\ f_{20}^{35}(C_{19} + \Delta_{19}, M) &= f_{20}^{35}(C_{19}, \tilde{M}) \end{aligned}$$

Thus by limiting the difference between two messages to a single word, the number of equations in the set of difference equations are reduced to a manageable size. However it should be noted that there does not necessarily exist solutions to such a set of difference equations.

8.2.2 Solution of Difference Equations

Once a set of difference equations is obtained, the problem remains to solve these equations. Dobbertin proposed one technique which resembles the operation of a genetic algorithm [14]. An investigation into the use of genetic algorithms to solve expressions similar to those encountered in the MD4 family are presented in [57]. The techniques used by Dobbertin to solve these sets of difference equations are summarised in [17] and [58]. The techniques required to solve these sets of equations are fairly specific to each set of equations. However, some common elements may be obtained from these techniques.

In both the analysis of MD4 and MD5 the difference between the messages are restricted to a single word. The Hamming distance between the two message words are kept small, since it is generally easier to manipulate the effect of a small number of bits on the chaining variables. The position at which the difference in the message word is introduced depends on the diffusion mechanisms of the hash function. One of the diffusion mechanisms in MD4 and MD5 is rotation. The bit positions in which differences are introduced are chosen specifically to counter the effect of rotation.

Another technique employed in both the attacks on MD4 and MD5 is the choice of specific values to simplify the sets of difference equations. These choices effectively allows the manipulation of the Boolean mappings. In the attack on MD5 this approach is used to establish certain differential patterns which simplifies the sets of difference equations.

The utilisation of a technique called iterative approximation is encountered in the attacks on MD4 and MD5. This technique evaluates the result obtained from a random initial setting. These initial settings are referred to as the basic values for the set of equations. The basic values are then changed by small increments and the corresponding result is observed. If the result is a closer approximation to the desired result, the changed basic values becomes the new basic values. This process is repeated until the desired result is obtained.

Another approach used to find solutions to sets of difference equations is used in the analysis of MD5. This technique requires that a large number of possible solutions are collected and then tried one after the other. This process is combined with the iterative approximation technique. A large number of basic values and possible solutions are generated. These possible solutions are then tried one after the other. The best solution is retained and serves as the basis for the next iteration. The techniques used to produce a large number of possible solutions are described in [57].

8.3 APPLICATION OF GENERALISED ATTACKS

The generalisation of the attack as described in this chapter was applied to a limited number of rounds of the HAVAL and SHA hash functions. Both are part of the MDx-family of hash functions. Using the generalised technique it can be shown that collisions can be establihed for limited rounds of HAVAL and single rounds of SHA. The details of these attacks are included in Chapters 9 and 10. To the best of my knowledge this is the first publicly published cryptanalytical results for SHA and HAVAL.

8.4 CONCLUSION

This chapter generalises the approach used by Dobbertin in the cryptanalysis of MD4 and MD5 as discussed in Chapters 6 and 7. The attacks on MD4 and MD5 focus on reducing the problem of finding a collision to solving a limited number of interrelated equations. This may be viewed as a divide and conquer approach. The complexity of finding collisions are reduced through two mechanisms. The first mechanism reduces the number of equations to be solved and the second simplifies these equations to facilitate the establishment of a solution. Detailed descriptions of these mechanisms may be found in [14], [17] and [57]. The application of this generalised technique to two other hash functions in the MD4 family, SHA-1 and HAVAL, is presented in the next two chapters.