Eindhoven University of Technology
Department of Mathematics and Computing Science

MASTER'S THESIS

On Collisions for MD5

By
M.M.J. Stevens

Supervisor:
Prof. dr. ir. H.C.A. van Tilborg

Advisors:
Dr. B.M.M. de Weger
Drs. G. Schmitz

Eindhoven, June 2007

Acknowledgements

I would like to express my gratitude to some people who were involved in this project. First of all, I owe thanks to Henk van Tilborg for being my overall supervisor and arranging this project and previous projects. I would like to thank Benne de Weger, who was especially involved in my work, for all his help, advice, comments, discussions, our joint work and his patience. The NBV deserve thanks for facilitating this project and I would like to thank Gido Schmitz especially for being my supervisor in the NBV. My gratitude goes out to Arjen Lenstra for comments, discussions, our joint work and my previous and future visits at EPFL. Thanks is due to Johan Lukkien for being on my committee.

This work benefited greatly from suggestions by Xiaoyun Wang. I am grateful for comments and assistance received from the anonymous Eurocrypt 2007 reviewers, Stuart Haber, Paul Hoffman, Pascal Junod, Vlastimil Klima, Bart Preneel, Eric Verheul, and Yiqun Lisa Yin. Furthermore, thanks go out to Jan Hoogma at LogicaCMG for technical discussions and sharing his BOINC knowledge and Bas van der Linden at TU/e for allowing us to use the Elegast cluster. Finally, thanks go out to hundreds of BOINC enthousiasts all over the world who donated an impressive amount of cpu-cycles to the HashClash project.

Contents

1 Introduction

This report is the result of my graduation project in completion of Applied Mathematics at the Eindhoven University of Technology (TU/e). It has been written in order to obtain the degree of Master of Science. The project has been carried out at the Nationaal Bureau Verbindings-beveiliging (NBV), which is part of the Algemene Inlichtingen en Veiligheids Dienst (AIVD) in Leidschendam.

1.1 Cryptographic hash functions

Hash functions are one-way functions with as input a string of arbitrary length (*the message*) and as output a fixed length string (*the hash value*). The hash value is a kind of signature for that message. One-way functions work in one direction, meaning that it is easy to compute the hash value from a given message and hard to compute a message that hashes to a given hash value.

They are used in a wide variety of security applications such as authentication, commitments, message integrity checking, digital certificates, digital signatures and pseudo-random generators. The security of these applications depend on the cryptographic strength of the underlying hash function. Therefore some security properties are required to make a hash function H suitable for such cryptographic uses:

P1. *Pre-image resistance*: Given a hash value h it should be hard to find any message m such that $h = H(m)$.

P2. *Second pre-image resistance*: Given a message m_1 it should be hard to find another message $m_2 \neq m_1$ such that $H(m_1) = H(m_2)$.

P3. *Collision resistance*: It should be hard to find different messages m_1, m_2 such that $H(m_1) = H(m_2)$.

A hash collision is a pair of different messages $m_1 \neq m_2$ having the same hash value $H(m_1) = H(m_2)$. Therefore second pre-image resistance and collision resistance are also known as weak and strong collision resistance, respectively. Since the domain of a hash function is much larger (can even be infinite) than its range, it follows from the pigeonhole principle that many collisions must exist. A brute force attack can find a pre-image or second pre-image for a general hash function with n-bit hashes in approximately 2^n hash operations. Because of the birthday paradox a brute force approach to generate collisions will succeed in approximately $2^{(n/2)}$ hash operations. Any attack that requires less hash operations than the brute force attack is formally considered a break of a cryptographical hash function.

Nowadays there are two widely used hash functions: MD5[17] and SHA-1[16]. Both are iterative hash functions based on the Merkle-Damgård[13, 1] construction and using a compression function. The compression function requires two fixed size inputs, namely a k-bit message block and a n-bit Intermediate Hash Value (internal state between message blocks denoted as IHV), and outputs the updated Intermediate Hash Value. In the Merkle-Damgård construction any message is first padded such that it has bitlength equal to a multiple of k and such that the last bits represent the original message length. The hash function then starts with a fixed IHV called the initial value and then updates IHV by applying the compression function with consecutive k-bit blocks, after which the IHV is returned as the n-bit hash value.

1.2 Collisions for MD5

MD5 (Message Digest algorithm 5) was designed by Ronald Rivest in 1991 as a strengthened version of MD4 with a hash size of 128 bits and a message block size of 512 bits. It is mainly based on 32-bit integers with addition and bitwise operations such as XOR, OR, AND and bitwise rotation. As an Internet standard, MD5 has been deployed in a wide variety of security applications and is also commonly used to check the integrity of files. In 1993, B. den Boer and A. Bosselaers[3] showed a weakness in MD5 by finding a "pseudo collision" for MD5 consisting of the same message with different initial values. H. Dobbertin[4] published in 1996 a semi free-start collision which consisted of two different 512-bit messages with a chosen initial value. This attack does not produce collisions for the full MD5, however it reveals that in MD5, differences in the higher order bits of the working state do not diffuse fast enough.

MD5 returns a hash value of 128 bits, which is small enough for a brute force birthday attack of order 2^{64}. Such a brute force attack was attempted by the distributed computing project MD5CRK which started in March 2004. However the project ended in August 2004 when Wang et al. [24] published their collisions for MD4, MD5, HAVAL-128 and RIPEMD, it is unknown to us how far the project was at that time. Later, Xiaoyun Wang and Hongbo Yu presented in [25] the underlying method to construct collisions using differential paths, which are a precise description how differences propagate through the MD5 compression function. However, they did so after Hawkes et al. [6] described in great detail a derivation of all necessary bitconditions on the working state of MD5 to satisfy the same differential paths.

The complexity of the original attack was estimated at 2^{39} calls to the compression function of MD5 and could be mounted in 15 minutes up to an hour on an IBM P690. Early improvements [26], [18], [12], [9] were able to find collisions in several hours on a single pc, the fastest being [9] which could find collisions for MD5 in about 2^{33} compressions.

Several results were published on how to abuse such collisions in the real world. The first were based only on the first published collision. In [7] it was shown how to achieve colliding archives, from which different contents are extracted using a special program. Similarly, in [14] a method was presented to construct two colliding files, both containing the same encrypted code, however only one file allows the possibly malicious code to be decrypted and executed by a helper program.

More complex applications use Wang's attack to find collisions starting and ending with some content, identical for both messages in the collision, specifically tailored to achieve a malicious goal. The most illustrative application is given by Daum and Lucks in [2] where they construct two colliding PostScript documents, each showing a different content. For other document formats, similar results can be achieved [5]. Also, the setting of digital certificates is not entirely safe as Lenstra and de Weger[11] presented two colliding X.509 certificates with different public keys, but with identical signatures from a Certificate Authority. Although as they contain the same identity there is no realistic abuse scenario.

1.3 Our Contributions

The contributions of this thesis are split into three main topics: speeding up collision finding, constructing differential paths and chosen-prefix collisions.

First we will show several improvements to speed up Wang's attack. All implementations of Wang's attack use bitconditions on the working state of MD5's compression function to find a message block which satisfies the differential path. We show how to find bitconditions on the working state such that differences are correctly rotated in the execution of the compression function, which was often neglected in collision finding algorithms and led to loss of efficiency. Also, in an analysis we show that the value of the IHV at the beginning of the attack has an impact on the complexity of collision finding. We give a recommendation to two bitconditions on this IHV to prevent a worst case complexity. Furthermore, we presented in [21], together with the above results, two new collision finding algorithms based on [9] which together allowed us to find collisions in about $2^{26.3}$ compressions for recommended IHV's. We were the first to present a method to find collisions in the order of one minute on a single pc, rather than hours. Later, Klima [10] gave another such method using a technique called Tunnels which was slightly faster, which we incorporated in our latest collision finding algorithm presented here. Currently, using also part of our second main result discussed below, we are able to find collisions for MD5 in about $2^{24.1}$ compressions for recommended IHV's which takes approx. 6 seconds on a 2.6Ghz Pentium4. Parts of our paper [21] were used in a book on applied cryptanalysis [20].

Wang's collision attack is based on two differential paths for the compression function which are to be used for consecutive message blocks where the first introduces differences in the IHV and the second eliminates these differences again. These two differential paths have been constructed by hand using great skill and intuition. However, an often posed question was how to construct differential paths in an automated way. In this thesis we present the first method to construct differential paths for the compression function of MD5. To show the practicality of our method we have constructed several new differential paths which can be found in the Appendix. Five of these differential paths were used to speedup Wang's attack as mentioned before. Our method even allows one to optimize the efficiency of the found differential paths for collision finding.

Our third contribution is the joint work with Arjen Lenstra and Benne de Weger in which we present a new collision attack on MD5, namely chosen-prefix collisions. A chosen-prefix collision consists of two arbitrarily chosen prefixes M and M' for which we can construct using our method two suffixes S and S', such that M extended with S and M' extended with S' collide under MD5: $MD5(M\|S) = MD5(M'\|S')$. Such chosen-prefix collisions allow more advanced abuse scenarios than the collisions based on Wang's attack. Using our method we have constructed an example consisting of two colliding X.509 certificates which (unlike in [11]) have different identities, but still receive the same signature from a Certification Authority. Although there is no realistic attack using our colliding certificates, this does constitute a breach of PKI principles. We discuss several other applications of chosen-prefix collisions which might be more realistic. This joint work [22] was accepted at EuroCrypt 2007 and has been chosen by the program committee to be one of the three notable papers which were invited to submit their work to the Journal of Cryptology.

1.4 Overview

In the following sections 2 and 3 we will fix some notation and give a definition of MD5 which we shall use throughout this thesis. Then we will describe the original attack on MD5 of Wang et al. in section 4. Our several improvements to speed up Wang's attack are presented in section 5. In section 6 we will discuss our method to construct differential paths for the compression function of MD5. Our joint work with Arjen Lenstra and Benne de Weger on chosen-prefix collisions and colliding certificates with different identities is presented in section 7. In section 8, we describe our use of the distributed computing framework BOINC in our project HashClash. Finally, we make some concluding remarks in section 9.

2 Preliminaries

MD5 operates on 32-bit unsigned integers called *words*, where we will number the bits from 0 (least significant bit) up to 31 (most significant bit). We use the following notation:

- Integers are denoted in hexadecimal together with a subscript 16, e.g. $\texttt{12ef}_{16}$,
  and in binary together with a subscript 2, e.g. $\texttt{0001001011101111}_2$,
  where the most significant digit is placed left;
- For words X and Y, addition $X + Y$ and substraction $X - Y$ are implicitly modulo 2^{32};
- $X[i]$ is the i-th bit of the word X;
- The cyclic left and right rotation of the word X by n bitpositions are denoted as $RL(X, n)$ and $RR(X, n)$, respectively:

$$\begin{aligned} & RL(\texttt{11110000011110010011110101001110}_2, 5) \\ = \;& \texttt{00001111001001111010100111001111}_2 \\ = \;& RR(\texttt{11110000011110010011110101001110}_2, 27); \end{aligned}$$

- $X \wedge Y$ is the bitwise AND of words X,Y or bits X,Y;
- $X \vee Y$ is the bitwise OR of words X,Y or bits X,Y;
- $X \oplus Y$ is the bitwise XOR of words X,Y or bits X,Y;
- $\overline{X}$ is the bitwise complement of the word or bit X;

A *binary signed digit representation* (BSDR) of a word X is a sequence $Y = (k_i)_{i=0}^{31}$, often simply denoted as $Y = (k_i)$, of 32 digits $k_i \in \{-1, 0, +1\}$ for $0 \leq i \leq 31$, where

$$X \equiv \sum_{i=0}^{31} k_i 2^i \mod 2^{32}, \qquad \text{e.g. } \texttt{fc00f000}_{16} \equiv (-1 \cdot 2^{12}) + (+1 \cdot 2^{16}) + (-1 \cdot 2^{26}).$$

Since there are 3^{32} possible BSDR's and only 2^{32} possible words, many BSDR's may exist for any given word X. For convenience, we will write BSDR's as a (unordered) sum of positive or negative powers of 2, instead of as a sequence, e.g. $-2^{12} + 2^{16} - 2^{26}$. This should not cause confusion, since it will always be clear from the context whether such a sum is a BSDR or a word.

The *weight* $w(Y)$ of a BSDR $Y = (k_i)$ is defined as the number of non-zero k_i's:

$$w(Y) = \sum_{i=0}^{31} |k_i|, \quad Y = (k_i);$$

We use the following notation for BSDR's:

- $Y \equiv X$ for a BSDR Y of the word X;
- $Y \equiv Y'$ for two BSDR's Y and Y' of the same word;
- $Y[\![i]\!]$ is the i-th signed bit of a BSDR Y;
- Cyclic left and right rotation by n positions of a BSDR Y is denoted as $RL(Y, n)$ and $RR(Y, n)$, respectively:

$$RL(-2^{31} + 2^{22} - 2^{10} + 2^0, 5) = -2^4 + 2^{27} - 2^{15} + 2^5.$$

A particularly useful BSDR of a word X which always exists is the Non-Adjacent Form (NAF), where no two non-zero k_i's are adjacent. The NAF is not unique since we work modulo 2^{32} (making $k_{31} = -1$ equivalent to $k_{31} = +1$), however we will enforce uniqueness of the NAF by choosing $k_{31} \in \{0, +1\}$. Among the BSDRs of a word, the NAF has minimal weight (see e.g. [15]).

3 Definition of MD5

A sequence of bits will be interpreted in a natural manner as a sequence of bytes, where every group of 8 consecutive bits is considered as one byte, with the leftmost bit being the most significant bit.

$$\text{E.g. } \texttt{01010011 11110000} = \texttt{01010011}_2\ \texttt{11110000}_2 = \texttt{53}_{16}\ \texttt{f0}_{16}$$

However, MD5 works on bytes using *Little Endian*, which means that in a sequence of bytes, the first byte is the least significant byte. E.g. when combining 4 bytes into a word, the sequence $\texttt{ef}_{16}$, $\texttt{cd}_{16}$, $\texttt{ab}_{16}$, $\texttt{89}_{16}$ will result in the word $\texttt{89abcdef}_{16}$.

3.1 MD5 Message Preprocessing

MD5 can be split up into these parts:

1. *Padding*:
   Pad the message with: first the '1'-bit, next as many '0' bits until the resulting bitlength equals $448 \mod 512$, and finally the bitlength of the original message as a 64-bit little-endian integer. The total bitlength of the padded message is $512N$ for a positive integer N.

2. *Partitioning*:
   The padded message is partitioned into N consecutive 512-bit blocks $M_1, M_2, \ldots, M_N$.

3. *Processing*:
   MD5 goes through $N+1$ states IHV_i, for $0 \leq i \leq N$, called the *intermediate hash values*. Each intermediate hash value IHV_i consists of four 32-bit words a_i, b_i, c_i, d_i. For $i = 0$ these are initialized to fixed public values:

   $$IHV_0 = (a_0, b_0, c_0, d_0) = (\texttt{67452301}_{16}, \texttt{EFCDAB89}_{16}, \texttt{98BADCFE}_{16}, \texttt{10325476}_{16}),$$

   and for $i = 1, 2, \ldots N$ intermediate hash value IHV_i is computed using the MD5 *compression function* described in detail below:

   $$IHV_i = \text{MD5Compress}(IHV_{i-1}, M_i).$$

4. *Output*:
   The resulting hash value is the last intermediate hash value IHV_N, expressed as the concatenation of the sequence of bytes, each usually shown in 2 digit hexadecimal representation, given by the four words a_N, b_N, c_N, d_N using Little-Endian. E.g. in this manner IHV_0 will be expressed as the hexadecimal string

   `0123456789ABCDEFFEDCBA9876543210`

3.2 MD5 compression function

The input for the compression function MD5Compress(IHV, B) is an intermediate hash value $IHV = (a, b, c, d)$ and a 512-bit message block B. There are 64 *steps* (numbered 0 up to 63), split into four consecutive *rounds* of 16 steps each. Each step uses a modular addition, a left rotation, and a non-linear function. Depending on the step t, an *Addition Constant* AC_t and a *Rotation Constant* RC_t are defined as follows, where we refer to Table A-1 for an overview of these values:

$$AC_t = \left\lfloor 2^{32} \left|\sin(t+1)\right| \right\rfloor , \quad 0 \leq t < 64,$$

$$(RC_t, RC_{t+1}, RC_{t+2}, RC_{t+3}) = \begin{cases} (7, 12, 17, 22) & \text{for } t = 0, 4, 8, 12, \\ (5, 9, 14, 20) & \text{for } t = 16, 20, 24, 28, \\ (4, 11, 16, 23) & \text{for } t = 32, 36, 40, 44, \\ (6, 10, 15, 21) & \text{for } t = 48, 52, 56, 60. \end{cases}$$

The non-linear function f_t depends on the round:

$$f_t(X,Y,Z) = \begin{cases} F(X,Y,Z) = (X \wedge Y) \oplus (\bar{X} \wedge Z) & \text{for } 0 \leq t < 16, \\ G(X,Y,Z) = (Z \wedge X) \oplus (\bar{Z} \wedge Y) & \text{for } 16 \leq t < 32, \\ H(X,Y,Z) = X \oplus Y \oplus Z & \text{for } 32 \leq t < 48, \\ I(X,Y,Z) = Y \oplus (X \vee \bar{Z}) & \text{for } 48 \leq t < 64. \end{cases}$$

The message block B is partitioned into sixteen consecutive 32-bit words $m_0, m_1, \ldots, m_{15}$ (using Little Endian byte ordering), and expanded to 64 words $(W_t)_{t=0}^{63}$ for each step using the following relations, see Table A-1 for an overview:

$$W_t = \begin{cases} m_t & \text{for } 0 \leq t < 16, \\ m_{(1+5t) \bmod 16} & \text{for } 16 \leq t < 32, \\ m_{(5+3t) \bmod 16} & \text{for } 32 \leq t < 48, \\ m_{(7t) \bmod 16} & \text{for } 48 \leq t < 64. \end{cases}$$

We follow the description of the MD5 compression function from [6] because its 'unrolling' of the cyclic state facilitates the analysis. For $t = 0, 1, \ldots, 63$, the compression function algorithm maintains a working register with 4 state words Q_t, Q_{t-1}, Q_{t-2} and Q_{t-3}. These are initialized as $(Q_0, Q_{-1}, Q_{-2}, Q_{-3}) = (b, c, d, a)$ and, for $t = 0, 1, \ldots, 63$ in succession, updated as follows:

$$\begin{aligned} F_t &= f_t(Q_t,\ Q_{t-1},\ Q_{t-2}), \\ T_t &= F_t + Q_{t-3} + AC_t + W_t, \\ R_t &= RL(T_t, RC_t), \\ Q_{t+1} &= Q_t + R_t. \end{aligned}$$

After all steps are computed, the resulting state words are added to the intermediate hash value and returned as output:

$$\text{MD5Compress}(IHV, B) = (a + Q_{61},\ b + Q_{64},\ c + Q_{63},\ d + Q_{62}).$$

4 MD5 Collisions by Wang et al.

X. Wang and H. Yu [25] revealed in 2005 their new powerful attack on MD5 which allowed them to find the collisions presented in 2004 [24] efficiently. A collision of MD5 consists of two messages and we will use the convention that, for an (intermediate) variable X associated with the first message of a collision, the related variable which is associated with the second message will be denoted by X'.

Their attack is based on a combined additive and XOR differential method. Using this differential they have constructed 2 *differential paths* for the compression function of MD5 which are to be used consecutively to generate a collision of MD5 itself. Their constructed differential paths describe precisely how differences between the two pairs (IHV, B) and (IHV', B'), of an intermediate hash value and an accompanying message block, propagate through the compression function. They describe the integer difference (-1, 0 or $+1$) in every bit of the intermediate working states Q_t and even specific values for some bits.

Using a *collision finding* algorithm they search for a collision consisting of two consecutive pairs of blocks (B_0, B_0') and (B_1, B_1'), satisfying the 2 differential paths which starts from arbitrary $I\hat{H}V = I\hat{H}V'$. Therefore the attack can be used to create two messages M and M' with the same hash that only differ slightly in two subsequent blocks as shown in the following outline where $I\hat{H}V = IHV_k$ for some k:

$$\begin{array}{ccccccccccccccc}
IHV_0 & \underset{M_1}{\rightarrow} & \cdots & \underset{M_k}{\rightarrow} & IHV_k & \underset{\mathbf{B_0}}{\rightarrow} & IHV_{k+1} & \underset{\mathbf{B_1}}{\rightarrow} & IHV_{k+2} & \underset{M_{k+3}}{\rightarrow} & \cdots & \underset{M_N}{\rightarrow} & IHV_N \\
= & & & & = & & \neq & & = & & & & = \\
IHV_0 & \underset{M_1}{\rightarrow} & \cdots & \underset{M_k}{\rightarrow} & IHV_k & \underset{\mathbf{B_0'}}{\rightarrow} & IHV'_{k+1} & \underset{\mathbf{B_1'}}{\rightarrow} & IHV'_{k+2} & \underset{M_{k+3}}{\rightarrow} & \cdots & \underset{M_N}{\rightarrow} & IHV_N
\end{array}$$

We will use this outline throughout this work with respect to this type of collisions. Note that all blocks $M_i = M_i'$ can be chosen arbitrarily and that only B_0, B_0', B_1, B_1' are generated by the collision finding algorithm.

This property was used in [11] to create two X.509 certificates where the blocks B_0, B_0', B_1, B_1' are embedded in different public keys. In [2] it was shown how to create two PostScript files with the same hash which showed two different but arbitrary contents.

The original attack finds MD5 collisions in about 15 minutes up to an hour on a IBM P690 with a cost of about 2^{39} compressions. Since then many improvements were made [18, 12, 26, 9, 21, 10]. Currently collisions for MD5 based on these differential paths can be found in several seconds on a single powerful pc using techniques based on *tunnels* [10], controlling rotations in the first round [21] and additional differential paths which we will present here.

4.1 Differential analysis

In [25] a combination of both integer modular substraction and XOR is used as differences, since the combination of both kinds of differences gives more information than each by themselves. So instead of only the integer modular difference between two related words X and X', this combination gives the integer differences (-1, 0 or $+1$) between each pair of bits $X[i]$ and $X'[i]$ for $0 \leq i \leq 31$. We will denote this difference as ΔX and represent it in a natural manner using BSDR's as follows

$$\Delta X = (k_i), \quad k_i = X'[i] - X[i] \ \text{ for } \ 0 \leq i \leq 31.$$

We will denote the regular modular difference as the word $\delta X = X' - X$ and clearly $\delta X \equiv \Delta X$.

As an example, suppose the integer modular difference is $\delta X = X' - X = 2^6$, then more than one XOR difference is possible:

- A one-bit difference in bit 6 ($X' \oplus X = \texttt{00000040}_{16}$) which means that $X'[6] = 1$, $X[6] = 0$ and $\Delta X = +2^6$.
- Two-bit difference in bits 6 and 7 caused by a carry. This happens when $X'[6] = 0$, $X[6] = 1$, $X'[7] = 1$ and $X[7] = 0$. Now $\Delta X = -2^6 + 2^7$.

- n-bit difference in bits 6 up to $6+n-1$ caused by $n-1$ carries. This happens when $X'[i]=0$ and $X[i]=1$ for $i=6,\dots,6+n-2$ and $X'[6+n-1]=1$ and $X[6+n-1]=0$. In this case $\Delta X = -2^6 - 2^7 \cdots - 2^{6+n-2} + 2^{6+n-1}$.

- A 26-bit difference in bits 6 up to 31 caused by 26 carries (instead of 25 as in the previous case). This happens when $X'[i]=0$ and $X[i]=1$ for $i=6,\dots,31$.

We extend the notation of δX and ΔX for a word X to any tuple of words coordinatewise. E.g. $\Delta IHV = (\Delta a, \Delta b, \Delta c, \Delta d)$ and $\delta B = (\delta m_i)_{i=0}^{15}$.

4.2 Two Message Block Collision

Wang's attack consists of two differential paths for two subsequent message blocks, which we will refer to as the first and second differential path. Although B_0 and B_1 are not necessarily the the first blocks of the messages M and M', we will refer to B_0 and B_1 as the first and second block, respectively. The first differential path starts with any given $IHV_k = IHV'_k$ and introduces a difference between IHV_{k+1} and IHV'_{k+1} which will be canceled again by the second differential path:

$$\delta IHV_{k+1} = (\delta a, \delta b, \delta c, \delta d) = (2^{31}, 2^{31}+2^{25}, 2^{31}+2^{25}, 2^{31}+2^{25}).$$

The first differential path is based on the following differences in the message block:

$$\delta m_4 = 2^{31}, \quad \delta m_{11} = 2^{15}, \quad \delta m_{14} = 2^{31}, \quad \delta m_i = 0,\ i \notin \{4, 11, 14\}$$

The second differential path is based on the *negated* message block differences:

$$\delta m_4 = -2^{31}, \quad \delta m_{11} = -2^{15}, \quad \delta m_{14} = -2^{31}, \quad \delta m_i = 0,\ i \notin \{4, 11, 14\}$$

Note that $-2^{31} = 2^{31}$ in words, so in fact δm_4 and δm_{14} are not changed by the negation.

These are very specific message block differences and were selected to ensure a low complexity for the collision finding algorithm as will be shown later.

4.3 Differential paths

The differential paths for both blocks (Tables B-1, B-2, see the Appendix) were constructed specifically to create a collision in this manner. The differential paths describe precisely for each of the 64 steps of MD5 what the differences are in the working state and how these differences pass through the boolean function and the rotation. More precisely, a differential path is defined through the sequences $(\delta m_t)_{t=0}^{15}$, $(\Delta Q_t)_{t=-3}^{64}$ and $(\delta T_t)_{t=0}^{64}$ of differences.

The first differential path starts without differences in the IHV, however differences will be introduced in step $t=4$ by δm_4. The second differential path starts with the given δIHV_{k+1}. In both, all differences in the working state will be canceled at step $t=25$ by δm_{14}. And from step $t=34$ both paths use the same differential steps, although with opposite signs. This structure can easily be seen in the Tables B-1 and B-2.

Below we show a fraction of the first differential path:

t	ΔQ_t	δF_t	δw_t	δT_t	RC_t
13	$-2^{24}+2^{25}+2^{31}$	$-2^{13}+2^{31}$	$-$	-2^{12}	12
14	$+2^{31}$	$2^{18}+2^{31}$	2^{31}	$2^{18}-2^{30}$	17
15	$+2^3-2^{13}+2^{31}$	$2^{25}+2^{31}$	$-$	$-2^7-2^{13}+2^{25}$	22
16	$-2^{29}+2^{31}$	2^{31}	$-$	2^{24}	5
17	$+2^{31}$	2^{31}	$-$	$-$	9
18	$+2^{31}$	2^{31}	2^{15}	2^3	14
19	$+2^{17}+2^{31}$	2^{31}	$-$	-2^{29}	20

The two differential paths were made by hand with great skill and intuition. It has been an open question for some time how to construct differential paths methodically. In section 6 we will present the first method to construct differential paths for MD5. Using our method we have constructed several differential paths for MD5. We use 5 differential paths in section 5 to speedup the attack by Wang et al. and 8 others were used in section 7 for a new collision attack on MD5.

4.4 Sufficient conditions

Wang et al. use *sufficient conditions* (modified versions are shown in Tables B-3,B-4) to efficiently search for message blocks for which these differential paths hold. These sufficient conditions guaranteed that the necessary carries and correct boolean function differences happen. Each condition gives the value of a bit $Q_t[i]$ of the working state either directly or indirectly as shown in Table 4-1. Later on we will generalize and extend these conditions to also include the value of the related bit $Q'_t[i]$.

Table 4-1: Sufficient bitconditions.

Symbol	condition on $Q_t[i]$	direct/indirect
.	none	direct
0	$Q_t[i] = 0$	direct
1	$Q_t[i] = 1$	direct
^	$Q_t[i] = Q_{t-1}[i]$	indirect
!	$Q_t[i] = \overline{Q_{t-1}[i]}$	indirect

These conditions are only to find a block B on which the message differences will be applied to find B' and should guarantee that the differential path happens. They can be derived for any differential path and there can be many different possible sets of sufficient conditions.

However, it should be noted that their sufficient conditions are not sufficient at all, as they do not guarantee that in each step the differences are rotated correctly. In fact as we will show later on, one does not want sufficient conditions for the full differential path as this increases the collision finding complexity significantly. On the other hand, sufficient conditions over the first round and necessary conditions for the other rounds will decrease the complexity. This can be seen as in the first round one can still choose the working state and one explicitly needs to verify the rotations, whereas in the other rounds the working state is calculated and verification can be done on the fly.

4.5 Collision Finding

Using these sufficient conditions one can efficiently search for a block B. Basically one can choose a random block B that meets all the sufficient conditions in the first round. The remaining sufficient conditions have to be fulfilled probabilistically and directly result in the complexity of this collision finding algorithm. Wang et al. used several improvements over this basic algorithm:

1. *Early abortion*:
   Abort at the step where the first sufficient condition fails.

2. *Multi-Message Modification*:
   When a certain condition in the second round fails, one can use multi-message modification. This is a substitution formula specially made for this condition on the message block B, such that after the substitution that condition will now hold without interfering with other previous conditions.

An example of multi-message modification is the following. When searching a block for the first differential path using Table B-3, suppose $Q_{17}[31] = 1$ instead of 0. This can be corrected by modifying m_1, m_2, m_3, m_4, m_5 as follows:

1. Substitute $\widehat{m_1} \leftarrow (m_1 + 2^{26})$, this results in a different $\widehat{Q_2}$.
2. Substitute $\widehat{m_2} \leftarrow (RR(Q_3 - \widehat{Q_2}, 17) - Q_{-1} - F(\widehat{Q_2}, Q_1, Q_0) - AC_2)$.
3. Substitute $\widehat{m_3} \leftarrow (RR(Q_4 - Q_3, 22) - Q_0 - F(Q_3, \widehat{Q_2}, Q_1) - AC_3)$.
4. Substitute $\widehat{m_4} \leftarrow (RR(Q_5 - Q_4, 7) - Q_1 - F(Q_4, Q_3, \widehat{Q_2}) - AC_4)$.
5. Substitute $\widehat{m_5} \leftarrow (RR(Q_6 - Q_5, 12) - \widehat{Q_2} - F(Q_5, Q_4, Q_3) - AC_5)$.

The first line is the most important, here m_1 is changed such that $\widehat{Q_{17}}[31] = 0$, assuming Q_{13} up to Q_{16} remain unaltered. The added difference $+2^{26}$ in m_1 results in an added difference of $+2^{31}$ in $Q_{17}[31]$, hence $\widehat{Q_{17}}[31] = 0$. The four other lines simply change m_2, m_3, m_4, m_5 such that Q_3 up to Q_{16} remain unaltered by the change in m_1. Since there are no conditions on Q_2, all previous conditions are left intact.

Wang et al. constructed several of such multi-message modifications which for larger t become more complex. Klima presented in [9] two collision finding algorithms, one for each block, which are much easier and more efficient than these multi-message modifications. Furthermore, Klima's algorithms work for arbitrary differential paths, while multi-message modifications have to be derived specifically for each differential path.

5 Collision Finding Improvements

In [6] a thorough analysis of the collisions presented by Wang et al. is presented. Not only a set of 'sufficient' conditions on Q_t, similarly as those presented in [25], is derived but also a set of necessary restrictions on T_t for the differential to be realized. These restrictions are necessary to correctly rotate the add-difference δT_t to δR_t. Collision finding can be done more efficiently by also satisfying the necessary restrictions on T_t used in combination with early abortion.

Fast collision finding algorithms as presented in [9] can choose message blocks B which satisfy the conditions for $Q_1, \ldots, Q_{16}$. As one can simply choose values of $Q_1, \ldots, Q_{16}$ fulfilling conditions and then calculate m_t for $t = 0, \ldots, 15$ using

$$m_t = RR(Q_{t+1} - Q_t, RC_t) - f_t(Q_t, Q_{t-1}, Q_{t-2}) - Q_{t-3} - AC_t.$$

Message modification techniques are used to change a block B such that $Q_1, \ldots, Q_{16}$ are changed slightly maintaining their conditions and that Q_{17} up to some Q_k do not change at all. Naturally, we want k to be as large as possible.

Although conditions for $Q_1, \ldots, Q_{16}$ can easily be fulfilled, this does not hold for the restrictions on T_t which still have to be fulfilled probabilistically. Our first collision finding improvement we present here is a technique to satisfy those restrictions on T_t using conditions on Q_t which can be satisfied when choosing a message block B.

The first block has to fulfill conditions of its differential path, however there are also conditions due to the start of the differential path of the second block. Although not immediately clear, the latter conditions have a probability to be fulfilled that depends on IHV_k, the intermediate hash value used to compress the first block. We will show this dependency and present two conditions that prevent a worst-case probability. The need for these two conditions can also be relieved with our following result.

Another improvement is the use of additional differential paths we have constructed using the techniques we will present in section 6. We present one differential path for the first block and 4 additional differential paths for the second block. The use of these will relax some conditions imposed on the first block due to the start of the differential path for the second block. As each of the now five differential paths for the second block has different conditions imposed on the first block, only one of those has to be satisfied to continue with the second block.

We were the first to present in [21] a collision finding algorithm which was able to find collisions for MD5 in the order of minutes on a single pc, based on Klima's algorithm in [9]. Shortly after, Klima presented in [10] a new algorithm which was slightly faster than ours using a technique called *tunneling*. We will explain this tunneling technique and present an improved version of our algorithm in [21] using this technique. These improvements in collision finding were crucial to our chosen-prefix construction, as the differential paths for chosen-prefix collisions usually have significantly more conditions than Wang's differential paths. Hence, the complexity to find collision blocks satisfying these differential paths is significantly higher (about 2^{42} vs. $2^{24.1}$ compressions).

Currently using these three improvements we are able to find collisions for MD5 in several seconds on a single pc (approx. 6 seconds on a 2.6Ghz Pentium4 pc). Source code and a windows executable can be downloaded from `http://www.win.tue.nl/hashclash/`.

5.1 Sufficient Conditions to control rotations

The first technique presented here allows to fulfill the restrictions on T_t by using extra conditions on Q_{t+1} and Q_t such as those in Table 4-1. By using the relation $Q_{t+1} - Q_t = R_t = RL(T_t, RC_t)$ we can control specific bits in T_t. In our analysis of Wang's differential paths, we searched for those restrictions on T_t with a significant probability that they are not fulfilled. For each such restriction on T_t, for $t = 0, \ldots, 19$, we have found bitconditions on Q_{t+1} and Q_t which were sufficient for the restriction to hold. For higher steps it is more efficient to directly verify the restriction instead of using conditions on Q_t.

All these restrictions can be found in [6] with a description why they are necessary for the differential path. The resulting conditions together with the original conditions can be found in

Table B-3. Below we will show the original set of sufficient conditions in [25] in black and our added conditions will be underlined and in blue.

5.1.1 Conditions on Q_t for block 1

1. **Restriction: $\Delta T_4 = -2^{31}$.**

   This restriction is necessary to guarantee that $\delta R_4 = -2^6$ instead of $+2^6$. The condition $T_4[31] = 1$ is necessary and sufficient for $\Delta T_4 = -2^{31}$ to happen. Bit 31 of T_4 is equal to bit 6 of R_4, since T_4 is equal to $RR(R_4, 7)$. By adding the conditions $Q_4[4] = Q_4[5] = 1$ and $Q_5[4] = 0$ to the conditions $Q_4[6] = Q_5[6] = 0$ and $Q_5[5] = 1$, it is guaranteed that $R_4[6] = T_4[31] = 1$. Satisfying other Q_t conditions, this also implies that $Q_6[4] = Q_5[4] = 0$.

$$\begin{array}{r|lr} Q_5[6-4] & \texttt{01}\underline{\texttt{0}}\ \cdots & \\ Q_4[6-4] & \texttt{0}\underline{\texttt{11}}\ \cdots & - \\ \hline R_4[6-4] & \texttt{11.}\ \cdots & = \end{array}$$

   This table shows the bits 4,5 and 6 of the words Q_5, Q_4 and R_4 with the most significant bit placed left, this is notated by $Q_5[6-4]$ extending the default notation for a single bit $Q_5[6]$.

2. **Restriction: add-difference -2^{14} in δT_6 must propagate to at least bit 15 on T_6.** This restriction implies that $T_6[14]$ must be zero to force a carry. Since $T_6[14] = R_6[31]$, the condition $T_6[14] = 0$ is guaranteed by the added conditions $Q_6[30-28, 26] = 0$. This also implies that $Q_5[30-28, 26] = 0$ because of other conditions on Q_t.

$$\begin{array}{r|lr} Q_7[31-23] & \texttt{000000111}\ \cdots & \\ Q_6[31-23] & \texttt{0}\underline{\texttt{000}}\texttt{0}\underline{\texttt{0}}\texttt{1.0}\ \cdots & - \\ \hline R_6[31-23] & \texttt{0000000..}\ \cdots & = \end{array}$$

   Note: in [26] these conditions were also found by statistical means.

3. **Restriction: add-difference $+2^{13}$ in δT_{10} must not propagate past bit 14 on T_{10}.** The restriction is satisfied by the condition $T_{10}[13] = R_{10}[30] = 0$. The conditions $Q_{11}[29-28] = Q_{10}[29] = 0$ and $Q_{10}[28] = 1$ are sufficient.

$$\begin{array}{r|lr} Q_{11}[31-28] & \texttt{00}\underline{\texttt{10}}\ \cdots & \\ Q_{10}[31-28] & \texttt{01}\underline{\texttt{11}}\ \cdots & - \\ \hline R_{10}[31-28] & \texttt{101.}\ \cdots & = \end{array}$$

4. **Restriction: add-difference -2^8 in δT_{11} must not propagate past bit 9 on T_{11}.** This restriction can be satisfied by the condition $T_{11}[8] = R_{11}[30] = 1$. With the above added condition $Q_{11}[29] = 1$ we only need the extra condition $Q_{12}[29] = 0$.

$$\begin{array}{r|lr} Q_{12}[31-29] & \texttt{00}\underline{\texttt{0}}\ \cdots & \\ Q_{11}[31-29] & \texttt{00}\underline{\texttt{1}}\ \cdots & - \\ \hline R_{11}[31-29] & \texttt{11.}\ \cdots & = \end{array}$$

5. **Restriction: add-difference -2^{30} in δT_{14} must not propagate past bit 31 on T_{14}.** For T_{14} the add difference -2^{30} must not propagate past bit 31, this is satisfied by either $T_{14}[30] = R_{14}[15] = 1$ or $T_{14}[31] = R_{14}[16] = 1$. This always happens when $Q_{15}[16] = 0$ and can be shown for the case if no carry from the lower order bits happens as well as the case if a negative carry does happen. A positive carry is not possible since we are subtracting.

no carry

$$\begin{array}{r|lr} Q_{15}[16-15] & \underline{\texttt{0}}\texttt{1}\ \cdots & \\ Q_{14}[16-15] & \texttt{11}\ \cdots & - \\ \hline R_{14}[16-15] & \texttt{10}\ \cdots & = \end{array}$$

negative carry from lower bits

$$\begin{array}{r|lr} Q_{15}[16-15] & \underline{\texttt{0}}\texttt{1}\ \cdots & \\ Q_{14}[16-15] & \texttt{11}\ \cdots & - \\ \hline R_{14}[16-15] & \texttt{01}\ \cdots & = \end{array}$$

6. **Restriction: add-difference -2^7 in δT_{15} must not propagate past bit 9 on T_{15}.** This can be satisfied by the added condition $Q_{16}[30] = \overline{Q_{15}[30]}$. Since then either $T_{15}[7] = R_{15}[29] = 1$, $T_{15}[8] = 1$ or $T_{15}[9] = 1$ holds. This can be shown if we distinguish between $Q_{15}[30] = 0$ and $Q_{15}[30] = 1$ and also distinguish whether or not a negative carry from the lower order bits happens.

no carry

$Q_{16}[31-29]$	`001`	$\cdots$	
$Q_{15}[31-29]$	`011`	$\cdots$	$-$
$R_{15}[31-29]$	`110`	$\cdots$	$=$

negative carry from lower bits

$Q_{16}[31-29]$	`001`	$\cdots$	
$Q_{15}[31-29]$	`011`	$\cdots$	$-$
$R_{15}[31-29]$	`101`	$\cdots$	$=$

no carry

$Q_{16}[31-29]$	`011`	$\cdots$	
$Q_{15}[31-29]$	`001`	$\cdots$	$-$
$R_{15}[31-29]$	`010`	$\cdots$	$=$

negative carry from lower bits

$Q_{16}[31-29]$	`011`	$\cdots$	
$Q_{15}[31-29]$	`001`	$\cdots$	$-$
$R_{15}[31-29]$	`001`	$\cdots$	$=$

7. **Restriction: add-difference $+2^{25}$ in δT_{15} must not propagate past bit 31 on T_{15}.** This is satisfied by the added condition $Q_{16}[17] = \overline{Q_{15}[17]}$. Since then either $T_{15}[25] = R_{15}[15] = 0$, $T_{15}[26] = 0$ or $T_{15}[27] = 0$ holds. We compactly describe all cases by mentioning which values were assumed for each result:

no carry

$Q_{16}[17-15]$	`!..`	$\cdots$		
$Q_{15}[17-15]$	`.01`	$\cdots$	$-$	
$R_{15}[17-15]$	`011`	$\cdots$	$=$	($Q_{16}[17-15]$ = `.00`)
	`100`	$\cdots$		($Q_{16}[17-15]$ = `.01`)
	`101`	$\cdots$		($Q_{16}[17-15]$ = `.10`)
	`110`	$\cdots$		($Q_{16}[17-15]$ = `.11`)

negative carry from lower bits

$Q_{16}[17-15]$	`!..`	$\cdots$		
$Q_{15}[17-15]$	`.01`	$\cdots$	$-$	
$R_{15}[17-15]$	`010`	$\cdots$	$=$	($Q_{16}[17-15]$ = `.00`)
	`011`	$\cdots$		($Q_{16}[17-15]$ = `.01`)
	`100`	$\cdots$		($Q_{16}[17-15]$ = `.10`)
	`101`	$\cdots$		($Q_{16}[17-15]$ = `.11`)

8. **Restriction: add-difference $+2^{24}$ in δT_{16} must not propagate past bit 26 on T_{16}.** This can be achieved with the added condition $Q_{17}[30] = \overline{Q_{16}[30]}$, since then always either $T_{16}[24] = R_{16}[29] = 0$ or $T_{16}[25] = R_{16}[30] = 0$.

no carry

$Q_{17}[30-29]$	`!.`	$\cdots$		
$Q_{16}[30-29]$	`.1`	$\cdots$	$-$	
$R_{16}[30-29]$	`01`	$\cdots$	$=$	($Q_{17}[30-29]$ = `00`)
	`10`	$\cdots$		($Q_{17}[30-29]$ = `01`)
	`01`	$\cdots$		($Q_{17}[30-29]$ = `10`)
	`10`	$\cdots$		($Q_{17}[30-29]$ = `11`)

negative carry from lower bits

$Q_{17}[30-29]$	`!.`	$\cdots$		
$Q_{16}[30-29]$	`.1`	$\cdots$	$-$	
$R_{16}[30-29]$	`00`	$\cdots$	$=$	($Q_{17}[30-29]$ = `00`)
	`01`	$\cdots$		($Q_{17}[30-29]$ = `01`)
	`00`	$\cdots$		($Q_{17}[30-29]$ = `10`)
	`01`	$\cdots$		($Q_{17}[30-29]$ = `11`)

9. **Restriction: add-difference -2^{29} in δT_{19} must not propagate past bit 31 on T_{19}.** This can be achieved with the added condition $Q_{20}[18] = \overline{Q_{19}[18]}$, since then always either $T_{19}[29] = 1$ or $T_{19}[30] = 1$.

no carry

$$\begin{array}{r|llll}
Q_{20}[18-17] & \texttt{!.} & \cdots & & \\
Q_{19}[18-17] & \texttt{.0} & \cdots & - & \\
\hline
R_{19}[18-17] & \texttt{10} & \cdots & = & (Q_{20}[18-17] = \texttt{00}) \\
 & \texttt{11} & \cdots & & (Q_{20}[18-17] = \texttt{01}) \\
 & \texttt{10} & \cdots & & (Q_{20}[18-17] = \texttt{10}) \\
 & \texttt{11} & \cdots & & (Q_{20}[18-17] = \texttt{11})
\end{array}$$

negative carry from lower bits

$$\begin{array}{r|llll}
Q_{20}[18-17] & \texttt{!.} & \cdots & & \\
Q_{19}[18-17] & \texttt{.0} & \cdots & - & \\
\hline
R_{19}[18-17] & \texttt{01} & \cdots & = & (Q_{20}[18-17] = \texttt{00}) \\
 & \texttt{10} & \cdots & & (Q_{20}[18-17] = \texttt{01}) \\
 & \texttt{01} & \cdots & & (Q_{20}[18-17] = \texttt{10}) \\
 & \texttt{10} & \cdots & & (Q_{20}[18-17] = \texttt{11})
\end{array}$$

10. **Restriction: add-difference $+2^{17}$ in δT_{22} must not propagate past bit 17 on T_{22}.** It is possible to satisfy this restriction with two Q_t conditions. However T_{22} will always be calculated in the algorithm we used, therefore it is better to verify directly that $T_{22}[17] = 0$. This restriction holds for both block 1 and 2.

11. **Restriction: add-difference $+2^{15}$ in δT_{34} must not propagate past bit 15 on T_{34}.** This restriction also holds for both block 1 and 2 and it should be verified with $T_{34}[15] = 0$.

5.1.2 Conditions on Q_t for block 2

Using the same technique as in the previous subsection we found 17 Q_t-conditions satisfying 12 T_t restrictions for block 2. An overview of all conditions for block 2 is included in Table B-4.

1. Restriction: $\Delta T_2[\![31]\!] = +1$.
   Conditions: $Q_1[16] = Q_2[16] = Q_3[15] = 0$ and $Q_2[15] = 1$.

2. Restriction: $\Delta T_6[\![31]\!] = +1$.
   Conditions: $Q_6[14] = 1$ and $Q_7[14] = 0$.

3. Restriction: $\Delta T_8[\![31]\!] = +1$.
   Conditions: $Q_8[5] = 1$ and $Q_9[5] = 0$.

4. Restriction: add-difference -2^{27} in δT_{10} must not propagate past bit 31 on T_{10}.
   Conditions: $Q_{10}[11] = 1$ and $Q_{11}[11] = 0$.

5. Restriction: add-difference -2^{12} in δT_{13} must not propagate past bit 19 on T_{13}.
   Conditions: $Q_{13}[23] = 0$ and $Q_{14}[23] = 1$.

6. Restriction: add-difference $+2^{30}$ in δT_{14} must not propagate past bit 31 on T_{14}.
   Conditions: $Q_{15}[14] = 0$.

7. Restriction: add-difference -2^{25} in δT_{15} must not propagate past bit 31 on T_{15}.
   Conditions: $Q_{16}[17] = \overline{Q_{15}[17]}$.

8. Restriction: add-difference -2^{7} in δT_{15} must not propagate past bit 9 on T_{15}.
   Conditions: $Q_{16}[28] = 0$.

9. Restriction: add-difference $+2^{24}$ in δT_{16} must not propagate past bit 26 on T_{16}.
   Conditions: $Q_{17}[30] = \overline{Q_{16}[30]}$.

10. Restriction: add-difference -2^{29} in δT_{19} must not propagate past bit 31 on T_{19}.
    Conditions: $Q_{20}[18] = \overline{Q_{19}[18]}$.

11. Restriction: add-difference $+2^{17}$ in δT_{22} must not propagate past bit 17 on T_{22}.
    See previous item 10.

12. Restriction: add-difference $+2^{15}$ in δT_{34} must not propagate past bit 15 on T_{34}.
    See previous item 11.

5.1.3 Deriving Q_t conditions

Deriving these conditions on Q_t to satisfy T_t restrictions can usually be done with a bit of intuition and naturally for step t one almost always has to look near bits 31 and RC_t of Q_t and Q_{t+1}. An useful aid is a program which, given conditions for $Q_1, \ldots, Q_{k+1}$, determines the probabilities of the correct rotations for each step $t = 1, \ldots, k$ and the joint probability that for steps $t = 1, \ldots, k$ all rotations are correct. The latter is important since the rotations affect each other.

Such a program could also determine extra conditions which would increase this joint probability. One can then look in the direction of the extra condition(s) that increases the joint probability the most. However deriving such conditions is not easily fully automated as the following two problems arise:

- Conditions guaranteeing the correct rotation of δT_t to δR_t may obstruct the correct rotation of δT_{t+1} to δR_{t+1}. Or even other δT_{t+k} for $k > 0$ if these conditions affect the values of Q_{t+k} and/or Q_{t+k+1} through indirect conditions.

- It is possible that to guarantee the correct rotation of some δT_t there are several solutions each consisting of multiple conditions. In such a case it might be that there is no single extra condition that would increase the joint probability significantly.

5.2 Conditions on the Initial Value for the attack

The intermediate hash value, IHV_k in the outline in section 4, used for compressing the first block of the attack, is called the initial value IV for the attack. This does not necessarily have to be the MD5 initial value, it could also result from compressing leading blocks. Although not completely obvious, the expected complexity and thus running time of the attack does depend on this initial value IV.

The intermediate value $IHV_{k+1} = (a_{k+1}, b_{k+1}, c_{k+1}, d_{k+1})$ resulting from the compression of the first block is used for compressing the second block and has the necessary conditions $c_{k+1}[25] = 1$ and $d_{k+1}[25] = 0$ for the second differential path to happen. The IHV_{k+1} depends on the $IV = (a, b, c, d)$ for the attack and $Q_{61}, \ldots, Q_{64}$ of the compression of the first block:

$$IHV_{k+1} = (a_{k+1}, b_{k+1}, c_{k+1}, d_{k+1}) = (a + Q_{61}, b + Q_{64}, c + Q_{63}, d + Q_{62}).$$

In [6] the sufficient conditions $Q_{62}[25] = 0$ and $Q_{63}[25] = 0$ are given. These conditions on $c_{k+1}[25]$ and $Q_{63}[25]$ can only be satisfied at the same time when

- either $c[25] = 1$ and there is no carry from bits 0-24 to bit 25 in the addition $c + Q_{63}$;

- or $c[25] = 0$ and there is a carry from bits 0-24 to bit 25 in the addition $c + Q_{63}$.

The conditions on $d_{k+1}[25]$ and $Q_{62}[25]$ can only be satisfied at the same time when

- either $d[25] = 0$ and there is no carry from bits 0-24 to bit 25 in the addition $d + Q_{62}$;

- or $d[25] = 1$ and there is a carry from bits 0-24 to bit 25 in the addition $d + Q_{62}$.

Satisfying all these conditions at the same time can even be impossible if for instance $c[25-0] = 0$, or $d[25] = 1 \wedge d[24-0] = 0$, since the necessary carry can never happen.

Luckily this doesn't mean the attack cannot be done for those IV's, since the conditions $Q_{62}[25] = 0$ and $Q_{63}[25] = 0$ are only sufficient. They allow the most probable differential path at those steps to happen, however there are other (less probable) differential paths that are also valid. If this normally most probable differential path cannot happen or happens with low probability (depending on the carry) then the average complexity of the attack depends on the probability that other differential paths happen. Experiments clearly indicated that the average runtime for this situation is significantly larger than the average runtime in the situation where the most probable differential path happens with high probability.

Therefore we relaxed all conditions on bit 25 of $Q_{60}, \ldots, Q_{63}$ to allow those other differential paths to happen. We also give a recommendation for the following two IV conditions to avoid this worst case:

$$c[25] = \overline{c[24]} \wedge d[25] = d[24] \quad \text{for} \quad IV = (a, b, c, d)$$

5.3 Additional Differential Paths

Furthermore, we have constructed new differential paths and conditions using the techniques we will present in section 6. We have constructed one differential path for the first block, which can be used as a replacement of the original first differential path.

We also have constructed four differential paths for the second block, each having different sets of conditions imposed on the first block. The first block only has to satisfy one of those sets of conditions. Then one can continue with the differential path for the second block that is associated with the satisfied set of conditions. Hence, together the five differential paths for the second block allow more freedom and improved collision finding for the first block.

Our differential paths for the first and second block were constructed using the exact same message block differences and IHV differences as the original first and second differential path, respectively. Also in step $t = 26$, ours and Wang's original differential paths have the same differences in the working state $(\delta Q_{26}, \delta Q_{25}, \delta Q_{24}, \delta Q_{23}) = (0, 0, 0, 0)$. Hence, also in later steps $t = 26, \ldots, 63$ our differential paths and conditions are equal to the respective original differential path and conditions.

Therefore we will omit steps $t = 26, \ldots, 63$ of our differential paths. We also applied conditions to control rotations using our technique in subsection 5.1. Our differential path for the first block is shown in Table B-5 and below, its conditions are shown in Table B-6. Our differential paths for the second block are shown in Table B-7, Table B-9, Table B-11 and Table B-13. The respective conditions are listed in Table B-8, Table B-10, Table B-12 and Table B-14.

Table 5-1: New first block differential path

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
$0-3$	$-$	$-$	$-$	$-$	$\cdot$
4	$-$	$-$	2^{31}	2^{31}	7
5	$-2^6 \ldots -2^{24}+2^{25}$	$-2^8+2^{14}-2^{19}$ $-2^{23}+2^{25}$	$-$	$-2^8+2^{14}-2^{19}$ $-2^{23}+2^{25}$	12
6	$+2^0-2^1+2^3-2^4$ $+2^5-2^6-2^7+2^8+2^{20}$ $+2^{21}-2^{22}+2^{26}-2^{31}$	$2^3-2^9+2^{15}$ $+2^{18}-2^{20}-2^{22}$	$-$	$2^3-2^9+2^{15}$ $+2^{18}-2^{20}-2^{22}$	17
7	-2^6+2^{31}	$-2^0+2^6-2^{10}$ $+2^{13}-2^{25}$	$-$	$-2^0+2^6-2^{10}$ $+2^{13}-2^{25}$	22
8	$-2^0+2^3-2^6-2^{15}$ $-2^{22}+2^{28}+2^{31}$	$-2^5+2^8+2^{15}$ $-2^{21}+2^{26}-2^{28}$	$-$	$2^5+2^8+2^{15}$ $-2^{21}+2^{26}-2^{28}$	7
9	$+2^0-2^6+2^{12}+2^{31}$	$-2^0+2^3-2^6+2^{31}$	$-$	$-2^1+2^5-2^{20}+2^{26}$	12
10	$-2^{12}+2^{17}+2^{31}$	$2^0-2^6+2^{12}+2^{31}$	$-$	$2^0-2^7+2^{12}$	17
11	$-2^{12}+2^{18}-2^{24}$ $+2^{29}+2^{31}$	$2^0-2^6-2^{17}$ $-2^{29}+2^{31}$	2^{15}	$2^3-2^7-2^{17}$ $-2^{22}-2^{28}$	22
12	$-2^7-2^{13}+2^{24}+2^{31}$	$2^7-2^{12}+2^{31}$	$-$	2^0+2^6	7
13	$+2^{24}+2^{31}$	2^{31}	$-$	$-2^{12}+2^{17}$	12
14	$+2^{29}+2^{31}$	$2^{24}+2^{29}+2^{31}$	2^{31}	$-2^{12}+2^{18}-2^{30}$	17
15	$+2^3-2^{15}-2^{31}$	$2^{24}+2^{31}$	$-$	$-2^7-2^{13}+2^{25}$	22
16	$-2^{29}-2^{31}$	2^{31}	$-$	2^{24}	5
17	-2^{31}	$-2^{29}+2^{31}$	$-$	$-$	9
18	-2^{31}	2^{31}	2^{15}	2^3	14
19	$+2^{17}-2^{31}$	2^{31}	$-$	-2^{29}	20
20	-2^{31}	2^{31}	$-$	$-$	5
21	-2^{31}	2^{31}	$-$	$-$	9
22	-2^{31}	2^{31}	$-$	2^{17}	14
23	$-$	$-$	2^{31}	$-$	20
24	$-$	2^{31}	$-$	$-$	5
25	$-$	$-$	2^{31}	$-$	9

5.4 Tunnels

In [10], Klima presented a new collision finding technique called *tunneling*. A tunnel allows one to make controlled changes in the message block B such that in Q_1 up to a certain Q_k, where k depends on the tunnel used, only small changes occur and all conditions remain unaffected. In fact, the effect of a tunnel is best shown using changes in a certain Q_m as we will show in the following example with $m = 9$ which is called the Q_9-tunnel.

5.4.1 Example: Q_9-tunnel

Assume that we have found a block B_0 that meets all first block conditions in Table B-3 up to Q_{24}. The conditions for Q_9, Q_{10} and Q_{11} are:

t	Conditions on Q_t: $b_{31} \ldots b_0$
9	`11111011 ...10000 0.1^1111 00111101`
10	`0111.... 0..11111 1101...0 01....00`
11	`0010.... ....0001 1100...0 11....10`

As this table shows, there are four bits in Q_9 that can be chosen freely, namely $Q_9[14]$, $Q_9[21]$, $Q_9[22]$ and $Q_9[23]$. If we change one of these bits, say $Q_9[22]$, without changing $Q_1, \ldots, Q_8$ and

$Q_{10}, \ldots, Q_{16}$ then only the following message block words are changed:

$$\begin{array}{lllllllllll}
m_8 & = & W_8 & = & RR(\mathbf{Q_9} - Q_8, 7) & - & f_8(Q_8, Q_7, Q_6) & - & Q_5 & - & AC_8 \\
m_9 & = & W_9 & = & RR(Q_{10} - \mathbf{Q_9}, 12) & - & f_9(\mathbf{Q_9}, Q_8, Q_7) & - & Q_6 & - & AC_9 \\
m_{10} & = & W_{10} & = & RR(Q_{11} - Q_{19}, 17) & - & f_{10}(Q_{10}, \mathbf{Q_9}, Q_8) & - & Q_7 & - & AC_{10} \\
m_{11} & = & W_{11} & = & RR(Q_{12} - Q_{11}, 22) & - & f_{11}(Q_{11}, Q_{10}, \mathbf{Q_9}) & - & Q_8 & - & AC_{11} \\
m_{12} & = & W_{12} & = & RR(Q_{13} - Q_{12}, 7) & - & f_{12}(Q_{12}, Q_{11}, Q_{10}) & - & \mathbf{Q_9} & - & AC_{12}
\end{array}$$

Hence, all conditions in the first round remain satisfied. In the second round Q_{17} and Q_{18} do not change, as steps $t = 16, 17$ do not depend on $m_8, \ldots, m_{12}$ as shown below:

Step t	16	17	18	19	20	21	22	23	24	25	26
Message block W_t	m_1	m_6	m_{11}	m_0	m_5	m_{10}	m_{15}	m_4	m_9	m_{14}	m_3
Affected Q_{t+1}	Q_{17}	Q_{18}	Q_{19}	Q_{20}	Q_{21}	Q_{22}	Q_{23}	Q_{24}	Q_{25}	Q_{26}	Q_{27}

On the other hand, a different m_{11} may lead to a different Q_{19}.

Suppose that $Q_{11}[22] = 1$ then

$$F_{11}[22] = f_{11}(Q_{11}[22], Q_{10}[22], Q_9[22]) = (Q_{11}[22] \wedge Q_{10}[22]) \oplus (\overline{Q_{11}[22]} \wedge Q_9[22]) = Q_{10}[22].$$

Hence F_{11} and thus also m_{11} do not change. In this case, actually Q_{17} up to Q_{21} remain unaffected by the change in $Q_9[22]$.

Furthermore, if we suppose that $Q_{10}[22] = 0$ then

$$F_{10}[22] = f_{10}(Q_{10}[22], Q_9[22], Q_8[22]) = (Q_{10}[22] \wedge Q_9[22]) \oplus (\overline{Q_{10}}[22] \wedge Q_8[22]) = Q_8[22]$$

and also m_{10} does not change. In this case we have achieved that a change in a single bit $Q_9[22]$ actually leaves Q_{17} up to Q_{24} unchanged and therefore all conditions in Q_1 up to Q_{24} remain satisfied.

In general, over multiple bits $Q_9[i_1], \ldots, Q_9[i_n]$ with $Q_{10}[i_1] = \ldots = Q_{10}[i_n] = 0$ and $Q_{11}[i_1] = \ldots = Q_{11}[i_n] = 1$, we find that changing those bits leads to a total of 2^n different message blocks, including the one we started with. And all those message blocks meet all conditions for Q_1 up to Q_{24}.

In the case of the first block conditions in Table B-3 we find that only bits $Q_9[21]$, $Q_9[22]$ and $Q_9[23]$ can be part of the Q_9-tunnel as $Q_{10}[14] = 1$ instead of 0. We need the extra conditions $Q_{10}[21] = Q_{10}[22] = 0$ and $Q_{11}[21] = Q_{11}[22] = Q_{11}[23] = 1$ to make use of this tunnel, as shown below in green and underlined.

t	Conditions on Q_t: $b_{31} \ldots b_0$
9	`11111011 xxx10000 0.1^1111 00111101`
10	`0111.... 00011111 1101...0 01....00`
11	`0010.... 111.0001 1100...0 11....10`

Initially the bits xxx should be set to `000` in a collision finding algorithm and when a message block B_0 is found that meets all conditions for Q_1 up to Q_{24} then we expand this B_0 into a set of 8 different message blocks using the 8 different values for these bits xxx. Q_{25} is the first affected Q_t for which we have to check if conditions are met, and is called the *point of verification* or POV. The number of bits that can be changed in a tunnel, in this case 3, is called the *strength* of the tunnel.

5.4.2 Notation for tunnels

We will use the notation $\mathcal{T}(Q_i, m_j)$ for the tunnel consisting of those bits of Q_i that do not change $W_{16}, \ldots, W_k$ but do change $W_{k+1} = m_j$. In other words those bits of Q_i that we can change such that $Q_{17}, \ldots, Q_{k+1}$ remain unaffected while Q_{k+2} does change. Naturally all such possible tunnels are disjoint as each bit of Q_i changes an unique first message word W_{k+1}. E.g. the example tunnel above consisting of the bits $Q_9[21]$, $Q_9[22]$ and $Q_9[23]$ and changing $W_{24} = m_9$ is notated as $\mathcal{T}(Q_9, m_9)$. Also since $Q_{10}[14] = 1$ the bit $Q_9[14]$ changes m_{10}, the bit $Q_9[14]$ is part of the tunnel $\mathcal{T}(Q_9, m_{10})$. Furthermore, the strength of a tunnel is the number of bits it consists of and is denoted as $\mathcal{S}_{i,j} = |\mathcal{T}(Q_i, m_j)|$.

The tunnels that we will use in our results are:

Table 5-2: Tunnels for collision finding

Tunnel	Required bitconditions	First affected Q_t, $t > 16$
$\mathcal{T}(Q_9, m_9)$	$Q_{10}[i] = 0 \wedge Q_{11}[i] = 1$	Q_{25}
$\mathcal{T}(Q_4, m_4)$	$Q_5[i] = 0 \wedge Q_6[i] = 1$	Q_{24}
$\mathcal{T}(Q_9, m_{10})$	$Q_{10}[i] = 1 \wedge Q_{11}[i] = 1$	Q_{22}
$\mathcal{T}(Q_{10}, m_{10})$	$Q_{11}[i] = 0$	Q_{22}
$\mathcal{T}(Q_4, m_5)$	$Q_5[i] = 1 \wedge Q_6[i] = 1$	Q_{21}
$\mathcal{T}(Q_5, m_5)$	$Q_6[i] = 0$	Q_{21}

It should be noted that the tunnels and their required bitconditions above depend only on the bits of Q_t and not on the bits of Q'_t. Below we show the different tunnel strengths for all differential paths in the Appendix:

Table 5-3: Tunnel strengths for known differential paths

Differential path	$\mathcal{S}_{9,9}$	$\mathcal{S}_{4,4}$	$\mathcal{S}_{9,10}$	$\mathcal{S}_{10,10}$	$\mathcal{S}_{4,5}$	$\mathcal{S}_{5,5}$	Total
Wang's first differential path	3	0	1	11	4	0	19
Wang's second differential path	9	6	2	3	0	1	21
Our first block diff. path	16	4	1	2	0	0	23
Our second block diff. path 1	9	0	3	2	0	0	15
Our second block diff. path 2	9	1	2	2	0	0	14
Our second block diff. path 3	9	0	2	3	0	1	15
Our second block diff. path 4	9	1	1	2	0	0	13
Our diff. path Table D-6	12	13	1	5	0	3	34
Our diff. path Table D-8	11	17	1	5	1	1	36
Our diff. path Table D-10	11	14	0	6	3	2	36
Our diff. path Table D-12	10	14	1	8	1	4	38
Our diff. path Table D-14	12	17	0	7	0	4	40
Our diff. path Table D-16	12	15	1	7	1	1	37
Our diff. path Table D-18	10	17	2	6	1	2	38
Our diff. path Table D-20	15	19	0	4	0	2	40

Especially in the last 8 differential paths above, one can see that we are able to optimize the tunnel strength when constructing differential paths.

5.5 Collision Finding Algorithm

In this section we will present our near-collision block search algorithm. It is an extension of our collision finding algorithms [21] shown here as Algorithm 5.1 and 5.2 which were again based on Klima's algorithms [9]. For each of the two collision blocks we used a separate collision finding algorithm. Using these two collision finding algorithms we were the first to be able to find collisions for MD5 in the order of minutes. Currently with our three improvements (conditions for the rotations, additional differential paths and the algorithms shown here) we are able to find collisions for MD5 in several seconds on a single pc.

These algorithms depend on the fact that given t, the message block word $W_t = m_k$ for some k can be calculated from Q_{t+1}, Q_t, Q_{t-1}, Q_{t-2}, Q_{t-3} using the formula

$$m_k = W_t = RR(Q_{t+1} - Q_t, RC_t) - f_t(Q_t, Q_{t-1}, Q_{t-2}) - Q_{t-3} - AC_t.$$

Hence, we can choose the working states for the first round satisfying their bitconditions and then determine the corresponding message block.

We extended these two collision finding algorithms using the tunnels in subsubsection 5.4.2. Furthermore we joined them into one near-collision block search algorithm in Algorithm 5.3 which also is suited for our differential paths we use later on (e.g. Table D-6). As these differential paths have a lot more bitconditions than the differential paths by Wang et al., we tried to maximize the number of choices at each step. During the construction of the differential paths themselves we also tried to maximize their total tunnel strength.

Using these optimizations we were able to efficiently find collision blocks for the differential paths we use later on (e.g. Table D-6) in chosen-prefix collisions using in the order of 2^{42} compressions, whereas using the basic algorithm in subsection 4.5 this would be infeasible. As these differential paths have a lot more bitconditions than e.g. the ones used in Wang's attack, the basic algorithm would need in the order of 2^{100} compressions to find a collision block, which is even harder than a brute-force collision search of approx. 2^{64} compressions.

Algorithm 5.1 Block 1 search algorithm

Note: conditions are listed in Table B-3. See subsection 5.1 for the conditions on T_{22} and T_{34}.

1. Choose $Q_1, Q_3, \ldots, Q_{16}$ fulfilling conditions;
2. Calculate $m_0, m_6, \ldots, m_{15}$;
3. Loop until $Q_{17}, \ldots, Q_{21}$ are fulfilling conditions:
   (a) Choose Q_{17} fulfilling conditions;
   (b) Calculate m_1 at $t = 16$;
   (c) Calculate Q_2 and m_2, m_3, m_4, m_5;
   (d) Calculate $Q_{18}, \ldots, Q_{21}$;
4. Loop over all possible Q_9, Q_{10} satisfying conditions such that m_{11} does not change:
   (Use tunnels $\mathcal{T}(Q_9, m_{10})$, $\mathcal{T}(Q_9, m_9)$ and $\mathcal{T}(Q_{10}, m_{10})$)
   (a) Calculate $m_8, m_9, m_{10}, m_{12}, m_{13}$;
   (b) Calculate $Q_{22}, \ldots, Q_{64}$;
   (c) Verify conditions on $Q_{22}, \ldots, Q_{64}, T_{22}, T_{34}$ and the IHV-conditions for the next block. Stop searching if all conditions are satisfied and a near-collision is verified.
5. Start again at step 1.

Algorithm 5.2 Block 2 search algorithm

Note: conditions are listed in Table B-4. See subsection 5.1 for the conditions on T_{22} and T_{34}.

1. Choose $Q_2, \dots, Q_{16}$ fulfilling conditions;
2. Calculate $m_5, \dots, m_{15}$;
3. Loop until $Q_{17}, \dots, Q_{21}$ are fulfilling conditions:
   (a) Choose Q_1 fulfilling conditions;
   (b) Calculate $m_0, \dots, m_4$;
   (c) Calculate $Q_{17}, \dots, Q_{21}$;
4. Loop over all possible Q_9, Q_{10} satisfying conditions such that m_{11} does not change:
   (Use tunnels $\mathcal{T}(Q_9, m_{10})$, $\mathcal{T}(Q_9, m_9)$ and $\mathcal{T}(Q_{10}, m_{10})$)
   (a) Calculate $m_8, m_9, m_{10}, m_{12}, m_{13}$;
   (b) Calculate $Q_{22}, \dots, Q_{64}$;
   (c) Verify conditions on $Q_{22}, \dots, Q_{64}, T_{22}, T_{34}$.
   Stop searching if all conditions are satisfied and a near-collision is verified.
5. Start again at step 1.

In our near-collision block search algorithm below in 5.3, one should keep the bits of tunnels $\mathcal{T}(Q_4, m_4)$, $\mathcal{T}(Q_4, m_5)$, $\mathcal{T}(Q_5, m_5)$, $\mathcal{T}(Q_9, m_9)$, $\mathcal{T}(Q_9, m_{10})$ and $\mathcal{T}(Q_{10}, m_{10})$ zero-valued. Only at the step where one uses the tunnel we will use the different values for the bits involved. It is more efficient to fix these tunnels before starting the collision search by applying their required conditions and making use of precomputed tables. However it is also possible to determine these tunnels at the step they are used. Furthermore, when e.g. using Wang's first block differential path one should not actually build the set $\mathcal{M}_0$ as all values of m_0 will do and 2^{32} words would require 16GB of memory. In general one should not build this set if it would require more memory than some large memory bound, and simply use random values m_0 at step 11. and then verify if Q_1 and Q_2 satisfy their conditions.

We have done a complexity analysis using our latest implementation of Wang's attack where we distinguish between three cases for the IV: the MD5 initial value IHV_0, recommended IV's as in subsection 5.2 and arbitrary IV's. Table 5-4 below shows the collision finding complexity as the cost equivalent to computing the stated number of compressions and the amount of time it takes on a 2.6Ghz Pentium4 pc.

Table 5-4: Collision finding complexity

IV case	Avg. complexity in compressions	Avg. time in seconds
MD5 $IV = IHV_0$	$2^{23.6}$	4.2
Recommended IV's	$2^{24.1}$	6.2
Random IV's	$2^{24.8}$	10.0

Algorithm 5.3 Near-collision block search algorithm

1. Choose random $Q_3, \ldots, Q_6$ and $Q_{13}, \ldots, Q_{17}$ fulfilling conditions;
2. Calculate m_1 at step $t = 16$;
3. Build a set $\mathcal{M}_0$ of values m_0 such that Q_1 and Q_2 resulting from m_0 and m_1 fulfill their conditions;
4. For all values of Q_7 that fulfill conditions do:
5. Calculate m_6 at step $t = 6$ and Q_{18} at step $t = 17$;
6. If Q_{18} does not satisfy conditions continue at step 4.;
7. For all values of $Q_8, \ldots, Q_{12}$ fulfilling conditions do:
8. Calculate m_{11} at step $t = 11$ and Q_{19} at step $t = 18$;
9. If Q_{19} does not satisfy conditions continue at step 7.;
10. For all $m_0 \in \mathcal{M}_0$ do:
11. Calculate Q_1, Q_2 and Q_{20} at steps $t = 0, 1, 19$ respectively;
12. If Q_{20} does not satisfy conditions continue at step 10.;
13. Use tunnels $\mathcal{T}(Q_4, m_5)$ and $\mathcal{T}(Q_5, m_5)$ and do:
14. Calculate m_5 at step $t = 5$ and Q_{21} at step $t = 20$;
15. If Q_{21} does not satisfy conditions continue at step 13.;
16. Use tunnels $\mathcal{T}(Q_9, m_{10})$ and $\mathcal{T}(Q_{10}, m_{10})$ and do:
17. Calculate m_{10} at step $t = 10$ and Q_{22} at step $t = 21$;
18. Calculate m_{15} at step $t = 15$ and Q_{23} at step $t = 22$;
19. If Q_{22} or Q_{23} does not satisfy conditions continue at step 16.;
20. Use tunnel $\mathcal{T}(Q_4, m_4)$, do:
21. Calculate m_4 at step $t = 4$ and Q_{24} at step $t = 23$;
22. If Q_{24} does not satisfy conditions continue at step 20.;
23. Use tunnel $\mathcal{T}(Q_9, m_9)$, do:
24. Calculate remaining m_i at $t = i \in \{0, \ldots, 15\}$;
25. Calculate $Q_{25}, \ldots, Q_{64}$;
26. Verify near-collision and return $B = (m_i)_{i=0}^{15}$ if so;
27. od; (step 23.)
28. od; (step 20.)
29. od; (step 16.)
30. od; (step 13.)
31. od; (step 10.)
32. od; (step 7.)
33. od; (step 4.)
34. Start again at step 1.

6 Differential Path Construction Method

Assume MD5Compress is applied to pairs of inputs for both intermediate hash value and message block, i.e., to (IHV, B) and (IHV', B'). We will assume that both δIHV and $\delta B = (\delta m_i)_{i=0}^{15}$ are given and possibly even IHV and IHV' or bits thereof. Note the slight abuse of notation here as we use only differences such as δm_i without specifying the values m_i and m_i'. We will continue to do so in our differential analysis.

A differential path for MD5Compress is a precise description of the propagation of differences through the 64 steps caused by δIHV and δB:

$$\begin{aligned}
\delta F_t &= f_t(Q_t', Q_{t-1}', Q_{t-2}') - f_t(Q_t, Q_{t-1}, Q_{t-2});\\
\delta T_t &= \delta F_t + \delta Q_{t-3} + \delta W_t;\\
\delta R_t &= RL(T_t', RC_t) - RL(T_t, RC_t);\\
\delta Q_{t+1} &= \delta Q_t + \delta R_t.
\end{aligned}$$

Note that δF_t is not uniquely determined by $\delta Q_t, \delta Q_{t-1}$ and δQ_{t-2}, so it is necessary to describe the value of δF_t and how it can result from the Q_i, Q_i' in such a way that it does not conflict with other steps. Similarly δR_t is not uniquely determined by δT_t and RC_t, so also the value of δR_t has to be described.

6.1 Bitconditions

We will use *bitconditions* on (Q_t, Q_t') to describe differential paths, where a single bitcondition specifies directly or indirectly the values of the bits $Q_t[i]$ and $Q_t'[i]$. Therefore, a differential path can be seen as a matrix of bitconditions with 68 rows (for the possible indices $t = -3, -2, \dots, 64$ in Q_t, Q_t') and 32 columns (one for each bit). A *direct* bitcondition on $(Q_t[i], Q_t'[i])$ does not involve other bits $Q_j[k]$ or $Q_j'[k]$, whereas an *indirect* bitcondition does, and specifically one of $Q_{t-2}[i]$, $Q_{t-1}[i]$, $Q_{t+1}[i]$ or $Q_{t+2}[i]$. Using only bitconditions on (Q_t, Q_t') we can specify all the values of δQ_t, δF_t and thus δT_t and $\delta R_t = \delta Q_{t+1} - \delta Q_t$ by the relations above. A bitcondition on $(Q_t[i], Q_t'[i])$ is denoted by $\mathfrak{q}_t[i]$, and symbols like `0`, `1`, `+`, `-`, `^`, ... are used for $\mathfrak{q}_t[i]$, as defined below. The 32 bitconditions $(\mathfrak{q}_t[i])_{i=0}^{31}$ are denoted by $\mathfrak{q}_t$. We discern between *differential* bitconditions and *boolean function* bitconditions. The former, shown in Table 6-1, are direct, and specify the

Table 6-1: Differential bitconditions.

$\mathfrak{q}_t[i]$	condition on $(Q_t[i], Q_t'[i])$	k_i
.	$Q_t[i] = Q_t'[i]$	0
+	$Q_t[i] = 0, \quad Q_t'[i] = 1$	$+1$
-	$Q_t[i] = 1, \quad Q_t'[i] = 0$	-1

Note: $\delta Q_t = \sum_{i=0}^{31} 2^i k_i$ and $\Delta Q_t = (k_i)$.

value $k_i = Q_t'[i] - Q_t[i]$ which together specify $\delta Q_t = \sum 2^i k_i$ by how each bit changes. Note that $\Delta Q_t = (k_i)$ is actually a BSDR of δQ_t. The boolean function bitconditions, shown in Table 6-2, are used to resolve any ambiguity in

$$\Delta F_t[\![i]\!] = f_t(Q_t'[i], Q_{t-1}'[i], Q_{t-2}'[i]) - f_t(Q_t[i], Q_{t-1}[i], Q_{t-2}[i]) \in \{-1, 0, +1\}$$

caused by different possible values for $Q_j[i], Q_j'[i]$ for given bitconditions.

As an example, for $t = 0$ and bitconditions $(\mathfrak{q}_t[i], \mathfrak{q}_{t-1}[i], \mathfrak{q}_{t-2}[i]) = (\texttt{.}, \texttt{+}, \texttt{-})$ there are two different possible values for the tuple $(Q_t[i], Q_t'[i], Q_{t-1}[i], Q_{t-1}'[i], Q_{t-2}[i], Q_{t-2}'[i])$ satisfying these bitconditions. As each case leads to a different boolean function difference, there is an ambiguity:

$$\begin{aligned}
\text{if } Q_t[i] = Q_t'[i] = 0 \text{ then } \Delta F_t[\![i]\!] &= f_t(0,1,0) - f_t(0,0,1) &= -1,\\
\text{but if } Q_t[i] = Q_t'[i] = 1 \text{ then } \Delta F_t[\![i]\!] &= f_t(1,1,0) - f_t(1,0,1) &= +1.
\end{aligned}$$

Table 6-2: Boolean function bitconditions.

$\mathtt{q}_t[i]$	condition on $(Q_t[i], Q'_t[i])$	direct/indirect	direction
0	$Q_t[i] = Q'_t[i] = 0$	direct	
1	$Q_t[i] = Q'_t[i] = 1$	direct	
^	$Q_t[i] = Q'_t[i] = Q_{t-1}[i]$	indirect	backward
v	$Q_t[i] = Q'_t[i] = Q_{t+1}[i]$	indirect	forward
!	$Q_t[i] = Q'_t[i] = \overline{Q_{t-1}[i]}$	indirect	backward
y	$Q_t[i] = Q'_t[i] = \overline{Q_{t+1}[i]}$	indirect	forward
m	$Q_t[i] = Q'_t[i] = Q_{t-2}[i]$	indirect	backward
w	$Q_t[i] = Q'_t[i] = Q_{t+2}[i]$	indirect	forward
#	$Q_t[i] = Q'_t[i] = \overline{Q_{t-2}[i]}$	indirect	backward
h	$Q_t[i] = Q'_t[i] = \overline{Q_{t+2}[i]}$	indirect	forward
?	$Q_t[i] = Q'_t[i] \wedge (Q_t[i] = 1 \vee Q_{t-2}[i] = 0)$	indirect	backward
q	$Q_t[i] = Q'_t[i] \wedge (Q_{t+2}[i] = 1 \vee Q_t[i] = 0)$	indirect	forward

To resolve this ambiguity, the bitconditions (.,+,-) can be replaced by either (0,+,-) or (1,+,-). Later on we will show how one can efficiently determine and resolve ambiguities methodically.

All boolean function bitconditions include the constant bitcondition $Q_t[i] = Q'_t[i]$, so they do not affect δQ_t. Furthermore, indirect boolean function bitconditions never involve a bit with condition + or -, since then it could be replaced by one of the direct bitconditions ., 0 or 1. We distinguish in the direction of indirect bitconditions, since that makes it easier to resolve an ambiguity later on. It is quite easy to change all backward bitconditions into forward ones in a valid (partial) differential pathm, and vice versa.

When all δQ_t and δF_t are determined by bitconditions then also δT_t and δR_t can be determined, which together describe the bitwise rotation of δT_t in each step. Note that this does not describe if it is a valid rotation or with what probability the rotation from δT_t to δR_t occurs.

6.2 Differential path construction overview

The basic idea in constructing a differential path is to construct a partial lower differential path over steps $t = 0, 1, \ldots, K$ for some K and a partial upper differential path over steps $t = K + 5, 17, \ldots, 63$, so that the Q_i involved in the partial paths meet but do not overlap. Then we will try to connect those partial paths over the remaining 4 steps into one full differential path. This will most likely fail and in general one will have to try to connect many pairs before finding a full valid differential path. The success probability depends heavily on the amount of freedom left by those bitconditions in the partial differential paths that affect the remaining steps $t = K+1, K+2, K+3, K+4$.

Connecting those two partial paths will result in a lot of bitconditions, hence it is best to have $K + 4 < 17$ to keep collision finding feasible. We chose $K = 12$ as then one can already determine (and maximize) the total tunnel strength of the resulting full differential path even before connecting. However, this choice may lead to problems as there can be a lot of conditions on $Q_{-2}, \ldots, Q_2$ and $Q_{13}, \ldots, Q_{17}$ which can result in a very limited (perhaps empty) set of values m_1 for which these conditions can simultaneously be satisfied. In this case, another good choice would be $K = 11$ as there one also has a good idea of total tunnel strength, however there will be less conditions on Q_{17} and more freedom for m_1.

Constructing the partial lower path can be done by starting with bitconditions $\mathtt{q}_{-3}$, $\mathtt{q}_{-2}$, $\mathtt{q}_{-1}$, $\mathtt{q}_0$ that are equivalent to given values of IHV, IHV' and then extend this step by step. Similarly a partial upper path can be constructed by extending the partial path in Table 7-1 step by step. Alternatively one can construct by hand any partial lower or upper differential path and then extend this step by step using our method. E.g. one could use the first and last parts of Wang's original differential paths and extend those till they meet and try to complete them in an effort to maximize the total tunnel strength.

To summarize, the algorithm for constructing a differential path consist of the following sub-steps:

1. Using IHV and IHV' determine bitconditions $(\mathfrak{q}_i)_{i=-3}^{0}$ which already form a partial lower differential path.
2. Generate a partial lower differential path by extending $(\mathfrak{q}_i)_{i=-3}^{0}$ forward up to step $t = K$.
3. Generate a partial upper differential path by extending the path in Table 7-1 down to $t = K + 5$.
4. Try to connect these lower and upper differential paths over $t = K+1, K+2, K+3, K+4$. If this fails generate other partial lower and upper differential paths and try again.

6.3 Extending partial differential paths

Suppose we have a partial differential path consisting of at least bitconditions $\mathfrak{q}_{t-1}$ and $\mathfrak{q}_{t-2}$ and that the values δQ_t and δQ_{t-3} are known. We assume that all indirect bitconditions are forward and do not involve bits of Q_t. We want to extend this partial differential path forward with step t resulting in the value δQ_{t+1} and (additional) forward bitconditions $\mathfrak{q}_t, \mathfrak{q}_{t-1}, \mathfrak{q}_{t-2}$ fulfilling our assumptions for the next step $t+1$. If we also have $\mathfrak{q}_t$ instead of only the value δQ_t (e.g. $\mathfrak{q}_0$ resulting from given values IHV, IHV'), then we can skip the carry propagation and continue at Section 6.3.2.

6.3.1 Carry propagation

First we want to use the value δQ_t to select bitconditions $\mathfrak{q}_t$. This can be done by choosing any BSDR of δQ_t, which directly translates into a possible choice for $\mathfrak{q}_t$ consisting of only differential bitconditions as given in Table 6-1. Since we want to construct differential paths with as few bitconditions as possible, but also want to be able to randomize the process, we may choose any low weight BSDR (such as the NAF).

6.3.2 Boolean function

For some i, let $(a, b, c) = (\mathfrak{q}_t[i], \mathfrak{q}_{t-1}[i], \mathfrak{q}_{t-2}[i])$ be any triple of bitconditions such that all indirect bitconditions involve only $Q_t[i]$, $Q_{t-1}[i]$ or $Q_{t-2}[i]$. The triple (a, b, c) is associated with the set U_{abc} of tuples of values $(x, x', y, y', z, z') = (Q_t[i], Q'_t[i], Q_{t-1}[i], Q'_{t-1}[i], Q_{t-2}[i], Q'_{t-2}[i])$:

$$U_{abc} = \left\{(x, x', y, y', z, z') \in \{0,1\}^6 \text{ satisfies bitconditions } (a,b,c)\right\}.$$

If $U_{abc} = \emptyset$ then (a, b, c) is said to be contradicting and cannot be part of any valid differential path. We define $\mathcal{F}_t$ as the set of all triples (a, b, c) such that all indirect bitconditions involve only $Q_t[i]$, $Q_{t-1}[i]$ or $Q_{t-2}[i]$ and $U_{abc} \neq \emptyset$.

We define V_{abc} as the set of all possible boolean function differences $\Delta F_t[\![i]\!] = f_t(x', y', z') - f_t(x, y, z)$ for given bitconditions $(a, b, c) \in \mathcal{F}_t$:

$$V_{abc} = \{f_t(x', y', z') - f_t(x, y, z) \mid (x, x', y, y', z, z') \in U_{abc}\} \subset \{-1, 0, +1\}.$$

There are bitconditions (d, e, f) such that $|V_{def}| = 1$, hence they leave no ambiguity and the triple (d, e, f) is said to be a *solution*. Let $\mathcal{S}_t \subset \mathcal{F}_t$ be the set of all solutions.

Now for arbitrary (a, b, c) and for each $g \in V_{abc}$ we define $W_{abc,g}$ as the set of solutions $(d, e, f) \in \mathcal{S}_t$ that are compatible with (a, b, c) and that have g as boolean function difference:

$$W_{abc,g} = \{(d, e, f) \in \mathcal{S}_t \mid U_{def} \subset U_{abc} \wedge V_{def} = \{g\}\}.$$

Note that for all $g \in V_{abc}$ there is always a triple $(d, e, f) \in W_{abc,g}$ that consists only of direct bitconditions `01+-` fixing a certain tuple in U_{abc}, hence $W_{abc,g} \neq \emptyset$. Even though $W_{abc,g}$ is not empty for all $g \in V_{abc}$, we are interested in bitconditions $(d,e,f) \in W_{abc,g}$ that maximizes $|U_{def}|$ as this maximizes the amount of freedom in the bits of Q_t, Q_{t-1} and Q_{t-2} while fixing $\Delta F_t[\![i]\!]$.

The direct and forward (resp. backward) boolean function bitconditions in Table 6-2 were chosen such that for all t, i and $(a,b,c) \in \mathcal{F}_t$ and for all $g \in V_{abc}$ there exists a triple $(d,e,f) \in W_{abc,g}$ consisting only of direct and forward (resp. backward) bitconditions such that

$$\{(x,x',y,y',z,z') \in U_{abc} \mid f_t(x',y',z') - f_t(x,y,z) = g\} = U_{def}.$$

In other words, the chosen boolean function bitconditions allows one to resolve an ambiguity in an optimal way.

If this triple $(d,e,f) \in W_{abc,g}$ is not unique, then we prefer direct over indirect bitconditions and short indirect bitconditions (`vy^!`) over long indirect bitconditions (`whqm#?`) for simplicity reasons. For given t, bitconditions (a,b,c), and $g \in V_{abc}$ we define $FC(t,abc,g) = (d,e,f)$ and $BC(t,abc,g) = (d,e,f)$ as the preferred triple $(d,e,f) \in W_{abc,g}$ consisting of direct and forward, respectively backward bitconditions satisfying

$$\{(x,x',y,y',z,z') \in U_{abc} \mid f_t(x',y',z') - f_t(x,y,z) = g\} = U_{def}.$$

These values can easily be determined and should be precomputed for all cases. Tables C-1, C-2, C-3 and C-4 show these values $FC(t,abc,g)$ and $BC(t,abc,g)$ for all t (grouped per boolean function) and all (a,b,c) consisting of differential bitconditions.

For all $i = 0,1,\ldots,31$ we have by assumption valid bitconditions $(a,b,c) = (\mathtt{q}_t[i], \mathtt{q}_{t-1}[i], \mathtt{q}_{t-2}[i])$ where only c can be an indirect bitcondition. If so, it must involve $Q_{t-1}[i]$. Therefore $(a,b,c) \in \mathcal{F}_t$. If $|V_{abc}| = 1$ there is no ambiguity and we let $\{g_i\} = V_{abc}$. Otherwise, if $|V_{abc}| > 1$, then we choose any $g_i \in V_{abc}$ and we resolve the ambiguity left by bitconditions (a,b,c) by replacing them by $(d,e,f) = FC(t,abc,g_i)$, which results in boolean function difference g_i.

Given all g_i, the values $\delta F_t = \sum_{i=0}^{31} 2^i g_i$ and $\delta T_t = \delta F_t + \delta Q_{t-3} + \delta W_t$ can be determined.

6.3.3 Bitwise rotation

The word δT_t does not uniquely determine the value of $\delta R_t = RL(T'_t,n) - RL(T_t,n)$, where $n = RC_t$. To determine a likely δR_t we use the fact that any BSDR (k_i) of δT_t fixes a δR_t:

$$\delta R_t = \sum_{i=0}^{31} 2^{i+n \bmod 32}(T'_t[i] - T_t[i]) = \sum_{i=0}^{31} 2^{i+n \bmod 32} k_i = 2^n \sum_{i=0}^{31-n} 2^i k_i + 2^{n-32} \sum_{i=32-n}^{31} 2^i k_i.$$

One can easily see that different BSDRs (k_i) and (l_i) of δT_t result in the same δR_t as long as

$$\sum_{i=0}^{31-n} 2^i k_i = \sum_{i=0}^{31-n} 2^i l_i \quad \text{and} \quad \sum_{i=32-n}^{31} 2^i k_i = \sum_{i=32-n}^{31} 2^i l_i.$$

In general, let $(\alpha,\beta) \in \mathbb{Z}^2$ be a partition of the word δT_t with $\alpha+\beta = \delta T_t \mod 2^{32}$, $|\alpha| < 2^{32-n}$, $|\beta| < 2^{32}$ and $2^{32-n}|\beta$. For any partition there is a BSDR (k_i) of δT_t such that

$$\alpha = \sum_{i=0}^{31-n} 2^i k_i \quad \text{and} \quad \beta = \sum_{i=32-n}^{31} 2^i k_i.$$

The converse also holds as for any BSDR (k_i) of δT_t defining α and β as above forms a partition (α,β) of δT_t. We will denote $(k_i) \equiv (\alpha,\beta)$ in this case.

The rotation of (α,β) is defined as

$$\delta R_t = RL((\alpha,\beta),n) = (2^n\alpha + 2^{n-32}\beta \mod 2^{32}) \quad (\equiv RL((k_i),n)).$$

This matches exactly the definition of rotating the BSDR (k_i). Clearly different partitions (α,β) of δT_t lead to different δR_t. We actually can describe all possible partitions quite easily and also determine their probability $Pr[\delta R_t = RL(X + \delta T_t, n) - RL(X,n)]$.

Let $x = (\delta T_t \mod 2^{32-n})$ and $y = (\delta T_t - x \mod 2^{32})$, then $0 \leq x < 2^{32-n}$ and $0 \leq y < 2^{32}$. This gives rise to at most 4 partitions of δT_t:

- $(\alpha, \beta) = (x, y)$;
- $(\alpha, \beta) = (x, y - 2^{32})$, if $y \neq 0$;
- $(\alpha, \beta) = (x - 2^{32-n}, y + 2^{32-n} \mod 2^{32})$, if $x \neq 0$;
- $(\alpha, \beta) = (x - 2^{32-n}, (y + 2^{32-n} \mod 2^{32}) - 2^{32})$, if $x \neq 0$ and $y + 2^{32-n} \neq 0 \mod 2^{32}$.

And these are all possible partitions of δT_t. The probability of each partition (α, β) equals

$$p_{(\alpha,\beta)} = \frac{2^{32-n} - |\alpha|}{2^{32-n}} \cdot \frac{2^{32} - |\beta|}{2^{32}}.$$

This formula is derived by counting the number of $0 \leq X < 2^{32}$ such that for the BSDR defined by $k_i = (X + \delta T_t)[i] - X[i]$ it holds that $(\alpha, \beta) \equiv (k_i)$. Looking only at the first $32 - n$ bits we can determine for a given α the probability that it will occur as $\alpha = \sum_{i=0}^{31-n} k_i$. This can be done by determining the number r of $0 \leq X < 2^{32-n}$ such that $0 \leq \alpha + X < 2^{32-n}$. Now we distinguish cases: if $\alpha < 0$ then $r = 2^{32-n} + \alpha$ and if $\alpha \geq 0$ then $r = 2^{32-n} - \alpha$. Hence $r = 2^{32-n} - |\alpha|$ out of 2^{32-n} X's. If $\alpha = \sum_{i=0}^{31-n} k_i$ holds then there is no carry to the higher bits and we can use the same argument for $\beta/2^{32-n}$. Hence, we conclude

$$p_{(\alpha,\beta)} = \frac{2^{32-n} - |\alpha|}{2^{32-n}} \cdot \frac{2^{n} - |\beta|2^{n-32}}{2^{n}} = \frac{2^{32-n} - |\alpha|}{2^{32-n}} \cdot \frac{2^{32} - |\beta|}{2^{32}}$$

One then chooses any partition (α, β) for which $p_{(\alpha,\beta)} \geq \frac{1}{4}$ and determines δR_t as $RL((\alpha, \beta), n)$. Previously in practice, we used $\delta R_t = RL(NAF(\delta T_t), n)$ as this often leads to the highest probability, especially given that we try to minimize the amount of differences in δQ_t and therefore also in δT_t and δR_t.

We would like to note that in previous work [19] a brute-force approach was used over all 2^{32} words X to find all possible $\delta R_t = RL(X + \delta T_t, n) - RL(X, n)$ resulting from δT_t and their probabilities. As we show here, finding all possible δR_t and their probabilities can be done very efficiently using a tiny number of computations.

6.4 Extending backward

Similar to extending forward, suppose we have a partial differential path consisting of at least bitconditions $\mathfrak{q}_t$ and $\mathfrak{q}_{t-1}$ and that the differences δQ_{t+1} and δQ_{t-2} are known. We want to extend this partial differential path backward with step t resulting in δQ_{t-3} and (additional) bitconditions $\mathfrak{q}_t, \mathfrak{q}_{t-1}, \mathfrak{q}_{t-2}$. We assume that all indirect bitconditions are backward and do not involve bits of Q_{t-2}.

We choose a BSDR of δQ_{t-2} with weight at most 1 or 2 above the lowest weight, such as the NAF. We translate the chosen BSDR into bitconditions $\mathfrak{q}_{t-2}$.

For all $i = 0, 1, \ldots, 31$ we have by assumption valid bitconditions $(a, b, c) = (\mathfrak{q}_t[i], \mathfrak{q}_{t-1}[i], \mathfrak{q}_{t-2}[i])$ where only b can be an indirect bitcondition. If so, it must involve $Q_{t-2}[i]$. Therefore $(a, b, c) \in \mathcal{F}_t$. If $|V_{abc}| = 1$ there is no ambiguity and we let $\{g_i\} = V_{abc}$. Otherwise, if $|V_{abc}| > 1$, then we choose any $g_i \in V_{abc}$ and we resolve the ambiguity left by bitconditions (a, b, c) by replacing them by $(d, e, f) = BC(t, abc, g_i)$, which results in boolean function difference g_i. Given all g_i, the value $\delta F_t = \sum_{i=0}^{31} 2^i g_i$ can be determined.

To rotate $\delta R_t = \delta Q_{t+1} - \delta Q_t$ over $n = 32 - RC_t$ bits, we simply choose a partition (α, β) of δR_t with probability $\geq 1/4$ and determine $\delta T_t = RL((\alpha, \beta), n)$. Finally, we determine $\delta Q_{t-3} = \delta T_t - \delta F_t - \delta W_t$ to extend our partial differential path backward with step t.

6.5 Constructing full differential paths

Construction of a full differential path can be done as follows. Choose δQ_{-3} and bitconditions $\mathfrak{q}_{-2}$, $\mathfrak{q}_{-1}$, $\mathfrak{q}_0$ and extend forward up to step 11. Also choose δQ_{64} and bitconditions $\mathfrak{q}_{63}, \mathfrak{q}_{62}, \mathfrak{q}_{61}$ and extend backward down to step 16. This leads to bitconditions $\mathfrak{q}_{-2}, \mathfrak{q}_{-1}, \ldots, \mathfrak{q}_{11}, \mathfrak{q}_{14}, \mathfrak{q}_{15}, \ldots, \mathfrak{q}_{63}$ and differences $\delta Q_{-3}, \delta Q_{12}, \delta Q_{13}, \delta Q_{64}$. It remains to finish steps $t = 12, 13, 14, 15$. As with extending backward we can, for $t = 12, 13, 14, 15$, determine δR_t, choose the resulting δT_t after right rotation of δR_t over RC_t bits, and determine $\delta F_t = \delta T_t - \delta W_t - \delta Q_{t-3}$.

We aim to find new bitconditions $\mathfrak{q}_{10}, \mathfrak{q}_{11}, \ldots, \mathfrak{q}_{15}$ that are compatible with the original bitconditions and that result in the required δQ_{12}, δQ_{13}, $\delta F_{12}, \delta F_{13}, \delta F_{14}, \delta F_{15}$, thereby completing the differential path. First we can test whether it is even possible to find such bitconditions.

For $i = 0, 1, \ldots, 32$, let $\mathcal{U}_i$ be a set of tuples $(q_1, q_2, f_1, f_2, f_3, f_4)$ of 32-bit integers with $q_j \equiv f_k \equiv 0 \mod 2^i$ for $j = 1, 2$ and $k = 1, 2, 3, 4$. We want to construct each $\mathcal{U}_i$ so that for each tuple $(q_1, q_2, f_1, f_2, f_3, f_4) \in \mathcal{U}_i$ there exist bitconditions $\mathfrak{q}_{10}[\ell], \mathfrak{q}_{11}[\ell], \ldots, \mathfrak{q}_{15}[\ell]$, determining the $\Delta Q_{11+j}[\![\ell]\!]$ and $\Delta F_{11+k}[\![\ell]\!]$ below, over the bits $\ell = 0, \ldots, i-1$, such that

$$\begin{aligned} \delta Q_{11+j} &= q_j + \sum_{\ell=0}^{i-1} 2^\ell \Delta Q_{11+j}[\![\ell]\!], \qquad && j = 1, 2, \\ \delta F_{11+k} &= f_k + \sum_{\ell=0}^{i-1} 2^\ell \Delta F_{11+k}[\![\ell]\!], \qquad && k = 1, 2, 3, 4. \end{aligned}$$

This implies $\mathcal{U}_0 = \{(\delta Q_{12}, \delta Q_{13}, \delta F_{12}, \delta F_{13}, \delta F_{14}, \delta F_{15})\}$. The other $\mathcal{U}_i$ are constructed inductively by Algorithm 6.1 by exhaustive search. Furthermore, $|\mathcal{U}_i| \leq 2^6$, since for each q_j, f_k there are at most 2 possible values that can satisfy the above relations.

If we find $\mathcal{U}_{32} \neq \emptyset$ then there exists a path $u_0, u_1, \ldots, u_{32}$ with $u_i \in \mathcal{U}_i$ where each u_{i+1} is generated by u_i in Algorithm 6.1. Now the desired new bitconditions $(\mathfrak{q}_{15}[i], \mathfrak{q}_{14}[i], \ldots, \mathfrak{q}_{10}[i])$ are $(a', b'', c''', d''', e'', f')$, which can be found at step 13 of Algorithm 6.1, where one starts with u_i and ends with u_{i+1}.

Clearly, the probability of success and thus the complexity of constructing a full differential path depends on several factors, where the amount of freedom left by the bitconditions $\mathfrak{q}_{10}, \mathfrak{q}_{11}, \mathfrak{q}_{14}, \mathfrak{q}_{15}$ and the number of possible BSDR's of δQ_{12} and δQ_{13} are the most important.

Algorithm 6.1 Construction of $\mathcal{U}_{i+1}$ from $\mathcal{U}_i$.

Suppose $\mathcal{U}_i$ is constructed as desired. Set $\mathcal{U}_{i+1} = \emptyset$ and for each tuple $(q_1, q_2, f_1, f_2, f_3, f_4) \in \mathcal{U}_i$ do the following:

1. Let $(a, b, e, f) = (\mathfrak{q}_{15}[i], \mathfrak{q}_{14}[i], \mathfrak{q}_{11}[i], \mathfrak{q}_{10}[i])$.
2. For each bitcondition $d = \mathfrak{q}_{12}[i] \in \begin{cases} \{\texttt{.}\} & \text{if } q_1[i] = 0 \\ \{\texttt{-}, \texttt{+}\} & \text{if } q_1[i] = 1 \end{cases}$ do
3. Let $q_1' = 0, -1, +1$ for resp. $d = \texttt{.}, \texttt{-}, \texttt{+}$
4. For each different $f_1' \in \{-f_1[i], +f_1[i]\} \cap V_{def}$ do
5. Let $(d', e', f') = FC(12, def, f_1')$
6. For each bitcondition $c = \mathfrak{q}_{13}[i] \in \begin{cases} \{\texttt{.}\} & \text{if } q_2[i] = 0 \\ \{\texttt{-}, \texttt{+}\} & \text{if } q_2[i] = 1 \end{cases}$ do
7. Let $q_2' = 0, -1, +1$ for resp. $c = \texttt{.}, \texttt{-}, \texttt{+}$
8. For each different $f_2' \in \{-f_2[i], +f_2[i]\} \cap V_{cd'e'}$ do
9. Let $(c', d'', e'') = FC(13, cd'e', f_2')$
10. For each different $f_3' \in \{-f_3[i], +f_3[i]\} \cap V_{bc'd''}$ do
11. Let $(b', c'', d''') = FC(14, bc'd'', f_3')$
12. For each different $f_4' \in \{-f_4[i], +f_4[i]\} \cap V_{ab'c''}$ do
13. Let $(a', b'', c''') = FC(15, ab'c'', f_4')$
14. Insert $(q_1 - 2^i q_1', q_2 - 2^i q_2', f_1 - 2^i f_1', f_2 - 2^i f_2', f_3 - 2^i f_3', f_4 - 2^i f_4')$ into $\mathcal{U}_{i+1}$.

Keep only one of each tuple in $\mathcal{U}_{i+1}$ that occurs multiple times. By construction we find $\mathcal{U}_{i+1}$ as desired.

7 Chosen-Prefix Collisions

A chosen-prefix collision is a pair of messages M and M' which consist of arbitrary chosen prefixes P and P' (not necessarily of the same length), together with constructed suffixes S and S' such that $M = P\|S$, $M' = P'\|S'$ and $MD5(M) = MD5(M')$. Furthermore, appending an arbitrary suffix S'' to each of these messages still leads to a collision $MD5(M\|S'') = MD5(M'\|S'')$ of MD5. In this section we will present our joint work with Arjen Lenstra and Benne de Weger which is a method to construct such chosen-prefix collisions. Using this method we have constructed one example of a chosen-prefix collision, namely two colliding X.509 certificates with different identities [22] which we will refer to often. Details on this example itself are discussed in subsection 7.5.

The two suffixes we will construct consist of three parts: padding bitstrings S_p and S'_p, followed by 'birthday' bitstrings S_b and S'_b, followed by 'near collision' blocks S_c and S'_c. The padding bitstrings S_p and S'_p are chosen to guarantee that the bitlengths of $P\|S_p$ and $P'\|S'_p$ are both equal to $L = 512n - 96$ for a positive integer n. They can be chosen arbitrarily but must meet the length requirements. The 'birthday' bitstrings S_b and S'_b both consist of 96 bits and complete the n-th block. Applying MD5 to $P\|S_p\|S_b$ and $P'\|S'_p\|S'_b$ will result in IHV_n and IHV'_n, respectively. The 'birthday' bitstrings are constructed in such a manner that δIHV_n can be eliminated using several near-collision blocks in S_c and S'_c as described below.

The main idea is to eliminate the difference δIHV_n using several consecutive near-collisions that together constitute S_c and S'_c. The number of differences in $\delta IHV_n = (\delta a, \delta b, \delta c, \delta d)$ is measured using the NAF weight, the total weight of the NAFs of δa, δb, δc and δd. For each near-collision we need to construct a differential path such that the NAF weight of the new δIHV_{n+j+1} is lower than the NAF weight of δIHV_{n+j}, until after r near-collisions we have reached $\delta IHV_{n+r} = (0, 0, 0, 0)$.

7.1 Near-collisions

We will use near-collisions based on a family of upper differential paths using the message block difference $\delta m_{11} = \pm 2^d$ for varying $0 \leq d \leq 31$ and $\delta m_i = 0$ for $i \neq 11$. This was suggested to us by Xiaoyun Wang as with this type of message difference the number of bitconditions over the final two rounds can be kept very low. This is illustrated in Table 7-1, where the corresponding upper differential path is shown for the final 31 steps. As one can see in Table A-1, these message block differences maximizes the number of steps in the third and fourth round with $\delta Q_t = 0$.

Table 7-1: Partial differential path with $\delta m_{11} = \pm 2^d$.

t	δQ_t	δF_t	δW_t	δT_t	δR_t	RC_t
30	$\mp 2^d$					
31	0					
32	0					
33	0	0	$\pm 2^d$	0	0	16
$34 - 60$	0	0	0	0	0	$\cdot$
61	0	0	$\pm 2^d$	$\pm 2^d$	$\pm 2^{d+10 \bmod 32}$	10
62	$\pm 2^{d+10 \bmod 32}$	0	0	0	0	15
63	$\pm 2^{d+10 \bmod 32}$	0	0	0	0	21
64	$\pm 2^{d+10 \bmod 32}$					

Although the number of bitconditions over the final two rounds is very low, the second round will contain in the order of 100 bitconditions. Would these bitconditions have occurred in the third or fourth round, they would have implied a collision finding complexity of approx. 2^{100} compressions. However, in our case there will be in the order of only 30 bitconditions from Q_{25} up to Q_{33}, where Q_{25} is the POV of the most efficient tunnel $\mathcal{T}(Q_9, m_9)$ (see Table 5-2). Because of this fact and using the collision finding techniques described in section 5, we were able to find actual near-collision blocks within feasible time.

7.2 Birthday Attack

The differential paths under consideration can only add (or substract) a tuple $(0, 2^i, 2^i, 2^i)$ to δIHV_{n+j} and therefore cannot eliminate arbitrary δIHV_n. Specifically, we need δIHV_n to be of the form $(0, \delta b, \delta b, \delta b)$ for some word δb.

To solve this we first use a birthday attack to find 'birthday' bitstrings S_b and S'_b such that $\delta IHV_n = (0, \delta b, \delta b, \delta b)$ for some δb. The birthday attack actually searches for a collision of $IHV_n = (a, b, c, d)$ and $IHV'_n = (a', b', c', d')$ such that $(a, b-c, b-d) = (a', b'-c', b'-d')$, implying indeed $\delta a = 0$ and $\delta b = \delta c = \delta d$. The search space consists of 96 bits, 3 words $(a, b-c, b-d)$ of 32 bits each, and therefore the birthday step can be expected to require on the order of $\sqrt{\frac{\pi}{2} 2^{96}} \approx 2^{49}$ calls to the MD5 compression function.

As soon as a collision with some δb is found, one can start eliminating the differences in δb. Using our family of upper differential paths we can eliminate any signed bit of δb. Since the NAF of δb has lowest weight among BSDR's, eliminating the signed bits in this NAF will lead to the lowest number of near-collisions required. Hence, on average one may expect to find a δb of NAF weight $32/3 \approx 11$. One may extend the birthdaying by searching for a δb of lower NAF weight. In the case of our colliding certificates example we found a δb of NAF weight only 8, after having extended the search somewhat longer than absolutely necessary.

When actually implementing such a birthday attack, one needs to fix a IHV selection function $\phi : (x, y, z) \mapsto \{IHV_n, IHV'_n\}$ and a message block generating function $\psi : (x, y, z) \mapsto B$. E.g. for ϕ one can use the parity of x to map either to IHV_n or IHV'_n and for ψ one can use a partial 416 bit block R and map to $R \| x \| y \| z$. These functions are used to compose the function $\Phi : (x, y, z) \mapsto (a, b-c, b-d)$ where $(a, b, c, d) = \mathrm{MD5Compress}(\phi(x, y, z), \psi(x, y, z))$, which is a deterministic pseudo-random walk in our 96 bit search space.

Applying generic Pollard-Rho, one can find a collision $\Phi(x, y, z) = \Phi(x', y', z')$ with $(x, y, z) \neq (x', y', z')$. The collision is useful only if $\phi(x, y, z) \neq \phi(x', y', z')$, i.e. the collision does not consist of only one of our chosen prefixes. Directly parallelizing Pollard-Rho using K instances does not lead to a factor K speedup, rather to a $\sqrt{K}$ speedup. We refer to [23] for a method to parallelize a birthday search leading to a factor K speedup. We have implemented this method in our birthday search for our chosen-prefix collision example.

Their general idea is to fix a relatively small set S of tuples (x, y, z) called distinguished points. E.g. all tuples (x, y, z) having $x = 0$. Each instance will generate 'trails' starting with a random (x_0, y_0, z_0) and iteratively calculate $(x_{i+1}, y_{i+1}, z_{i+1}) = \Phi(x_i, y_i, z_i)$ until a distinguished point $(x_l, y_l, z_l) \in S$ is reached. Each trail can be stored using only its starting point (x_0, y_0, z_0), its ending point $(x_l, y_l, z_l) \in S$ and its length l. When one trail meets another trail in a point then the two trails will coincide from that point on and will end in the same distinguished point. Hence, a collision is detected when different trails result in the same distinguished point. The collision itself can then be found by recalculating both trails to the point where they meet first.

However, there are some small issues one has to be aware of. When a trail reaches its starting point it will fall into an endless cycle without ever reaching a distinguished point. To avoid this case one should abort any trail whose length exceeds a certain limit, e.g. a limit set to 20 times the expected trail length. It is also possible that a trail reaches the starting point of another trail so that both end in the same distinguished point without yielding an actual collision. This cannot be avoided and should only occur with a very small probability.

7.3 Iteratively Reducing IHV-differences

Assume we have found birthday bitstrings such that $\delta IHV_n = (0, \delta b, \delta b, \delta b)$ and let (k_i) be the NAF of δb. Then we can reduce $\delta IHV_n = (0, \delta b, \delta b, \delta b)$ to $(0, 0, 0, 0)$ by using, for each non-zero k_i, a differential path based on the partial differential path in Table 7-1 with $\delta m_{11} = -k_i 2^{i-10 \bmod 32}$. In other words, the signed bit difference at position i in δb can be eliminated by choosing a message difference only in δm_{11}, with just one opposite-signed bit set at position $i - 10 \bmod 32$. Let i_j for $j = 1, 2, \ldots, r$ be the indices of the non-zero k_i. Starting with n-block messages $M = P \| S_\mathrm{p} \| S_\mathrm{b}$ and $M' = P' \| S'_\mathrm{p} \| S'_\mathrm{b}$ and the corresponding resulting IHV_n and IHV'_n we do the following for

$j = 1, 2, \ldots, r$ in succession:

1. Let $\delta M_{n+j} = (\delta m_i)$ where $\delta m_{11} = -k_{i_j} 2^{i_j - 10 \bmod 32}$ and $\delta m_\ell = 0$ for $\ell \neq 11$.
2. Find a full differential path as shown in section 6 by connecting a lower differential path starting from IHV_{n+j-1} and IHV'_{n+j-1} and an upper differential path based on Table 7-1.
3. Find message blocks $S_{c,j}$ and $S'_{c,j} = S_{c,j} + \delta M_{n+j}$, that satisfy the differential path using the techniques shown in section 5.
4. Let $IHV_{n+j} = \text{MD5Compress}(IHV_{n+j-1}, S_{c,j})$, $IHV'_{n+j} = \text{MD5Compress}(IHV'_{n+j-1}, S'_{c,j})$, and append $S_{c,j}$ to M and $S'_{c,j}$ to M'.

After r iterations we will have found a chosen-prefix collision consisting of $M = P \| S_p \| S_b \| S_c$ and $M' = P' \| S'_p \| S'_b \| S'_c$ where S_c and S'_c consist of the r near-collision blocks $S_c = S_{c,1} \| \cdots \| S_{c,r}$ and $S'_c = S'_{c,1} \| \cdots \| S'_{c,r}$ just found. Any suffix S_s appended to both messages $M \| S_s$, $M' \| S_s$ will still lead to a full collision of MD5, which is useful to construct meaningful collisions for MD5.

7.4 Improved Birthday Search

The following partial differential path is a variant of Table 7-1 using the same message block differences. They differ only in the very last step where an additional bitdifference occurs. Both partial differential paths have almost the same probability, one never differing more than a factor 2 from the other. If we also incorporate the use of this variant upper differential path then we

Table 7-2: Variant partial differential path with $\delta m_{11} = \pm 2^d$.

t	δQ_t	δF_t	δW_t	δT_t	δR_t	RC_t
30	$\mp 2^d$					
31	0					
32	0					
33	0	0	$\pm 2^d$	0	0	16
$34-60$	0	0	0	0	0	$\cdot$
61	0	0	$\pm 2^d$	$\pm 2^d$	$\pm 2^{d+10 \bmod 32}$	10
62	$\pm 2^{d+10 \bmod 32}$	0	0	0	0	15
63	$\pm 2^{d+10 \bmod 32}$	0	0	0	0	21
64	$\pm 2^{d+10 \bmod 32} \mp 2^{d+31 \bmod 32}$					

can eliminate any $\delta IHV_n = (\delta a, \delta b, \delta c, \delta d)$ of the form $\delta a = 0$, $\delta c = \delta d$. Note that there is no limitation on δb which corresponds to δQ_{64}.

A strategy eliminating the differences in a δIHV_n of that form using near-collisions based on the differential paths in Table 7-1 and Table 7-2, denoted as DP_1 and DP_2 respectively, is the following. Let $\delta e = \delta b - \delta c$ and consider $(v_i) = RR(NAF(-\delta c), 10)$ and $(w_i) = RR(NAF(\delta e), 31)$. Then a non-zero v_i corresponds to a bitdifference in δb, δc, δd that can be eliminated using DP_1 with $\delta m_{11} = v_i 2^i$ as shown in the previous subsection. Similarly a non-zero w_i corresponds to a difference in $\delta b - \delta d$, i.e. one of the extra differences we allowed, which can be eliminated with DP_2 using $\delta m_{11} = w_i 2^i$. In the latter case one still has to deal with a corresponding difference in δb, δc, δd as we show below.

As a trivial example, suppose $\delta IHV = (0, +2^{12} - 2^1, +2^{12}, +2^{12})$. This clearly can be eliminated using DP_2 with $\delta m_{11} = -2^2$ as also the BSDR's (v_i) and (w_i) indicate:

$$(v_i) = RR(NAF(-2^{12}), 10) = RR(-2^{12}, 10) = -2^2,$$

$$(w_i) = RR(NAF(-2^1), 31) = RR(-2^1, 31) = -2^2.$$

Depending on the values of v_i and w_i for each bit $i = 0, \ldots, 31$ we can eliminate the corresponding bitdifferences in δIHV_n with either 1 or 2 near-collision blocks. There are five distinct cases which we analyze below:

1. When $v_i = 0$ and $w_i = 0$ there is no difference to be eliminated.
2. Suppose $v_i \neq 0$ and $w_i = 0$, then we can use DP_1 with $\delta m_{11} = v_i 2^i$ as before to eliminate the corresponding bitdifferences.
3. Suppose $v_i = w_i \neq 0$ then we can use DP_2 with $\delta m_{11} = v_i 2^i$ to eliminate the corresponding bitdifferences as shown in the example.
4. Suppose $v_i = 0$ and $w_i \neq 0$ then we can use one near-collision based on DP_2 with $\delta m_{11} = w_i 2^i$. This introduces a new difference $w_i 2^{i+10 \mod 32}$ in δb, $\delta c = \delta d$, which we correct using a second near-collision based on DP_1 with $\delta m_{11} = -w_i 2^i$.
5. Suppose $v_i \neq 0$ and $w_i = -v_i$. In this case we use DP_2 with $\delta m_{11} = w_i 2^i$. As in the previous case this introduces the bitdifference $w_i 2^{i+10 \mod 32}$ in δb, $\delta c = \delta d$. As $v_i = -w_i$ this signed bitdifference was already present in δb and $\delta c = \delta d$ and a carry happens. If $i + 10 = 31$ then this carry is lost and both differences v_i and w_i are eliminated. However if $i + 10 \neq 31$ then we can eliminate this carry bitdifference using DP_1 with $\delta m_{11} = v_i 2^{i+1 \mod 32}$.

As in the previous section we use DP_1 and DP_2 with a given δm_{11} and the current IHV_{n+j-1} and IHV'_{n+j-1} to construct a full differential path. Making use of our collision finding algorithm we find message blocks $S_{c,j}$ and $S'_{c,j}$ satisfying this differential path. We append these message blocks to M and M', respectively, and continue with the resulting IHV_{n+j} and IHV'_{n+j} until $\delta IHV = (0, 0, 0, 0)$.

Given that (v_i) and (w_i) are rotated NAF's, the probability that a signed bit v_i or w_i is nonzero equals 1/3. Also, v_i or w_i equals a specific value +1 or −1 with probability 1/6. Hence, we can determine the probability for each of the five cases above:

Case	1	2	3	4	5
Probability	$\frac{2}{3}^2 = \frac{4}{9}$	$\frac{1}{3} \cdot \frac{2}{3} = \frac{2}{9}$	$\frac{1}{3} \cdot \frac{1}{6} = \frac{1}{18}$	$\frac{2}{3} \cdot \frac{1}{3} = \frac{2}{9}$	$\frac{1}{3} \cdot \frac{1}{6} = \frac{1}{18}$
# Near-collisions	0	1	1	2	2

The expected number of required near-collisions per bit is $(\frac{2}{9} + \frac{1}{18}) \cdot 1 + (\frac{2}{9} + \frac{1}{18}) \cdot 2 = \frac{5}{6}$. It follows that we can expect to need $\frac{5}{6} \cdot 32 \approx 27$ near-collision blocks to eliminate all differences in a random δIHV_n of the form $\delta a = 0$ and $\delta c = \delta d$.

The birthday search has to be slightly modified as we only need a 64-bit search space. As before, we need a IHV selection function $\phi : (x, y) \mapsto \{IHV_n, IHV'_n\}$ and a message block generating function $\psi : (x, y) \mapsto B$. These functions are used to compose the function $\Phi : (x, y) \mapsto (a, c - d)$ where $(a, b, c, d) = \text{MD5Compress}(\phi(x, y), \psi(x, y))$. When a birthday collision $\Phi(x, y) = \Phi(x', y')$ with $\phi(x, y) \neq \phi(x', y')$ occurs, we have found message blocks which result in a δIHV of the required form $\delta a = 0$ and $\delta c = \delta d$.

This more advanced strategy has not been tried, however we intend to construct another chosen-prefix collision using this strategy in future work. One can also optimize between birthday complexity and the number of required near-collision blocks. Finding a single birthday collision costs $\sqrt{\frac{\pi}{2} 2^{64}} \approx 2^{33}$ compressions which is much more feasible compared to the previous birthday search. One can easily extend the birthday search, as the cost for subsequent birthday collisions decreases, to find collisions with fewer required near-collision blocks. An experimentation indicated that the cost of finding a collision requiring approx. 14 near-collision blocks is approx. 2^{39} compressions.

7.5 Colliding Certificates with Different Identities

In March 2005 it was shown how Wangs collisions could be used to construct two different valid and unsuspicious X.509 certificates with identical digital signatures [11]. These two colliding certificates differed only in the two collision blocks which were hidden in the RSA moduli. In particular, their Distinguished Name fields containing the identities of the certificate owners were equal.

It would be interesting to be able to select Distinguished Name fields which are different and chosen at will, non-random and human readable as one would expect from these fields. This can be realized now as in our chosen-prefix collisions one can extend two arbitrarily chosen messages such that the extended message collide. To achieve identical digital signatures for X.509 certificates one does not need to construct full certificates which collide under MD5, rather only the to-be-signed parts of the certificates need to collide under MD5.

We have constructed such an example of colliding X.509 certificates with different Distinguished Name fields where the suffixes S_b and S_c are hidden in the first half of the RSA moduli. The second half of the RSA moduli was constructed as in [11] to complete the RSA moduli n_1 and n_2 in such a manner that both are the product of two large primes and that the full certificates still collide under MD5.

7.5.1 To-be-signed parts

The to-be-signed parts up to the first bit of the RSA moduli were carefully constructed to have equal bitlength with the last block exactly 96 bits short of a full block. These to-be-signed parts consist of several fields compliant with the X.509 standard and the ASN.1 DER encoding rules.

We actually constructed three chosen-prefixes to increase the probability that $\phi(x,y,z) \neq \phi(x',y',z')$ when a birthday collision $\Phi(x,y,z) = \Phi(x',y',z')$ is found. Naturally we continued with only two of the three chosen-prefixes after the birthday search. The three chosen-prefixes have Distinguished Names "Arjen K. Lenstra", "Marc Stevens" and "Benne de Weger", notated as P_{AL}, P_{MS} and P_{BW} respectively. The chosen-prefixes are given as bitstrings in Table D-1, Table D-2 and Table D-3. Below we list all fields, and their values, which are contained in the encoded chosen-prefixes:

Field 1. X.509 version number: *Version 3* and identical for all three certificates;

Field 2. Serial number: Different in each chosen-prefix:

$$\begin{aligned} P_{\text{AL}}: &\quad \texttt{010c0001}_{16}, \\ P_{\text{MS}}: &\quad \texttt{020c0001}_{16}, \\ P_{\text{BW}}: &\quad \texttt{030c0001}_{16}; \end{aligned}$$

Field 3. Signature algorithm: *md5withRSAEncryption* for all chosen-prefixes;

Field 4. Issuer Distinguished Name: The Certificate Authority (CA) and identical in each case:

CN (Common Name)	=	"Hash Collision CA",
L (Locality)	=	"Eindhoven",
C (Country)	=	"NL";

Field 5. Validity period: Our certificates have the same validity period:

Not before	:	Jan. 1, 2006, 00h00m01s GMT
Not after	:	Dec. 31, 2007, 23h59m59s GMT

Field 6. Subject Distinguished Name: The identities are different in the Common Name (CN) and Organisation (O) fields for each certificate: (The organisation name is chosen such that the CN and O fields together hold exactly 29 characters to meet the length requirements on the chosen-prefixes.)

P_{AL}	P_{MS}	P_{BW}
CN = "Arjen K. Lenstra"	CN = "Marc Stevens"	CN="Benne de Weger"
O = "Collisionairs"	O="Collision Factory"	O="Collisionmakers"
L="Eindhoven"	L="Eindhoven"	L="Eindhoven"
C="NL"	C="NL"	C="NL"

Field 7. Public key algorithm: *rsaEncryption* for all chosen-prefixes;

Field 8. RSA modulus: Only the length specifier of the RSA modulus is part of the chosen-prefixes and is set to 8192 bits. The first byte after each chosen-prefix is also the first byte of the RSA modulus itself.

When we have found the RSA moduli we only need to complete the to-be-signed parts with the following fields and compute the digital signature of the CA using the MD5 hash of the colliding to-be-signed parts:

Field 9. RSA exponent: $\texttt{010001}_{16} = 65537$;

Field 10. Version 3 extensions: We use default values for these extensions:

$$\begin{aligned} \text{Basic Constraints}: &\quad \text{End Entity (not an CA), no limit on certification path length} \\ \text{Key Usage}: &\quad \text{Digital Signature, Non-Repudiation, Key Encipherment} \end{aligned}$$

7.5.2 Chosen-Prefix Collision Construction

Each of these chosen-prefixes consist of three full message blocks, resulting in some IHV_3, and one partial message block R of 416 bits which is identical for all three prefixes. We denote the three different IHV_3's as IHV_{AL}, IHV_{MS} and IHV_{BW} for prefixes P_{AL}, P_{MS} and P_{BW}, respectively. There is very limited space in a RSA modulus of 8192 bit and we also need enough freedom to complete the RSA moduli as a product of two large primes. Therefore we chose to use the original birthday search in subsection 7.2.

Given the three IHV's and R we defined the pseudo-random walk in the 96-bit search space as follows:

$$\phi(x,y,z) = \begin{cases} IHV_{\text{AL}}, & \text{if } x = 0 \mod 3; \\ IHV_{\text{MS}}, & \text{if } x = 1 \mod 3; \\ IHV_{\text{BW}}, & \text{if } x = 2 \mod 3. \end{cases}$$

$$\begin{aligned} \psi(x,y,z) &= R\|x\|y\|z \\ \rho(IHV) = \rho(a,b,c,d) &= (a, d-b, d-c) \\ \Phi(x,y,z) &= \rho(\text{MD5Compress}(\phi(x,y,z), \psi(x,y,z))) \end{aligned}$$

So given a 96-bit value (x,y,z) we use it to complete the message block R, determine which IHV_3 to use and compute the resulting IHV_4. We map this $IHV_4 = (a,b,c,d)$ to the 96-bit search space as $(a, d-b, d-c)$ as then a collision implies $\delta a = 0$, $\delta b = \delta c = \delta d$. We used the method of distinguished points to parallelize the birthday search where we defined the set of distinguished points as:

$$S = \{(x,y,z) \mid (x \equiv 0 \mod 2^{15}) \wedge (RL(y,15) \equiv 0 \mod 2^{15})\}.$$

Our birthday search resulted in a total of 120 collisions of which 80 were useful (different IHV's). We chose the following birthday collision as it requires only 8 near-collisions to eliminate the resulting δIHV_4:

$$(X,Y,Z) = (\texttt{cbb4091a}_{16},\ \texttt{7a26c740}_{16},\ \texttt{9b7f01af}_{16})$$

$$(X',Y',Z') = (\texttt{d6e773ee}_{16},\ \texttt{ba4fb3b3}_{16},\ \texttt{023d39a1}_{16})$$

This birthday collision gives us birthday bitstrings $S_b = X\|Y\|Z$ and $S'_b = X'\|Y'\|Z'$ which are appended to P_{MS} and P_{AL}, respectively, as $\phi(X,Y,Z) = IHV_{\text{MS}}$ and $\phi(X',Y',Z') = IHV_{\text{AL}}$. The extended chosen-prefixes $P_{\text{MS}}\|S_b$ and $P_{\text{AL}}\|S'_b$ consist of exactly four message blocks and result in $\delta IHV_4 = (0, \delta b_4, \delta b_4, \delta b_4)$ where

$$\delta b_4 = -2^5 - 2^7 - 2^{13} + 2^{15} - 2^{18} - 2^{22} + 2^{26} - 2^{30}.$$

We eliminated these bitdifferences in δIHV_4 with 8 consecutive near-collision blocks based on the differential path in Table 7-1.

As outlined before, we construct a full differential path starting with IHV_4 and IHV'_4 and using Table 7-1 with $\delta m_{11} = +2^{20}$ to eliminate -2^{30} in δb_4. The differential path we have found is shown in Table D-6 in the Appendix. The near-collision blocks M_5, M'_5 satisfying this differential path and the resulting IHV_5, IHV'_5 that we have found are shown in Table D-7. The other differences were eliminated similarly using the values -2^{16}, $+2^{12}$, $+2^8$, -2^5, $+2^3$, $+2^{29}$ and $+2^{27}$ for δm_{11} in that order. The differential paths we have constructed using these values for δm_{11} and the near-collision blocks $M_6, M'_6, \ldots, M_{12}, M'_{12}$ we found which satisfying them are shown in Tables D-8 up to D-21.

The birthday bitstrings S_b, S'_b and the 8 near-collisions blocks together form S_c, S'_c and are the $96 + 8 \times 512 = 4192$ most-significant bits of the RSA moduli. Using the method described in [11] we have found a bitstring S_m such that $S_b \| S_c \| S_m$ and $S'_b \| S'_c \| S_m$ form RSA moduli n_1 and n_2, respectively, as products of two large primes. The bitstring S_m and the smallest primes dividing n_1 and n_2 are given in the Appendix in Table D-24 and Table D-25.

We completed the to-be-signed parts using identical suffixes for both messages (including S_m) after the chosen-prefix collision $P_{\mathrm{MS}} \| S_b \| S_c$ and ${}_{\mathrm{AL}} \| S'_b \| S'_c$, hence the resulting to-be-signed parts collide under MD5. These certificates have identical signatures and can be found at our website: `http://www.win.tue.nl/hashclash/TargetCollidingCertificates/`.

7.5.3 Attack Scenarios

Though our colliding certificates construction involving different identities should have more attack potential than the one with identical identities in [11], we have not been able to find truly convincing attack scenarios. The core of PKI is to provide a relying party with trust, beyond reasonable cryptographic doubt, that the person belonging to the identity in the certificate has exclusive control over the private key corresponding to the public key in the certificate. Ideally, a realistic attack should attack this core of PKI and also enable the attack to cover his trails.

However, our construction requires that the two colliding certificates are generated simultaneously. Although each resulting certificate by itself is completely unsuspicious, the fraud becomes apparent when the two certificates are put alongside, as may happen during a fraud analysis.

Another problem is that the attacker must have sufficient control over the CA to predict all fields appearing before the public key, such as the serial number and the validity periods. It has frequently been suggested that this is an effective countermeasure against colliding certificate constructions in practice, but there is no consensus how hard it is to make accurate predictions. When this condition of sufficient control over the CA by the attacker is satisfied, colliding certificates based on chosen-prefix collisions are a bigger threat than those based on random collisions.

Obviously, the attack becomes effectively impossible if the CA adds a sufficient amount of fresh randomness to the certificate fields before the public key, such as in the serial number (as some already do, though probably for different reasons). This randomness is to be generated after the approval of the certification request. On the other hand, in general a relying party cannot verify this randomness. In our opinion, trustworthiness of certificates should not crucially depend on such secondary and circumstantial aspects. On the contrary, CAs should use a trustworthy hash function that meets the design criteria. Unfortunately, this is no longer the case for MD5.

7.6 Other Applications

7.6.1 Colliding Documents

Entirely different abuse scenarios are also possible. In [2] it was shown how to construct a pair of PostScript files which collide under MD5, and that show different messages to output media such as screen or printer. Similar constructions for several other document formats are presented in [5]. However, in those constructions both messages had to be hidden in each of the colliding files, which obviously raises suspicions upon inspection at bit level.

This can be avoided using chosen-prefix collisions. For example, two different messages can be entered into a document format which allows insertion of color images (such as PostScript, Adobe PDF, Microsoft Word), with one message per document. Each document can be constructed carefully with at the last page a color image containing constructed birthday and near-collision bitstrings such that the documents collide under MD5. The image itself can be a short one pixel wide line, or hidden inside a layout element, a company logo, or in the form of a nicely colored barcode claiming to be some additional security feature, obviously offering far greater security than those old-fashioned black and white barcodes.

Figure 1: The example chosen-prefix collision built into bitmap images.

In Figure 1 the actual 4192-bit collision-causing appendages computed for the certificates are built into bitmaps to get two different barcode examples. Each string of 4192 bits leads to one line of 175 pixels, say A and B, and the barcodes consist of the lines ABBBBB and BBBBBB respectively. Apart from the 96 most significant bits corresponding to the 4 pixels in the upper left corner, the barcodes differ in only a few bits, which makes the resulting color differences hard to spot for the human eye.

7.6.2 Misleading Integrity Checking

In [14] and [7] it was shown how to abuse existing MD5 collisions to mislead integrity checking software based on MD5. Similar to the colliding Postscript applications, they also used the differences in the colliding inputs to construct deviating execution flows of some programs.

Here too, chosen-prefix collisions allow a more elegant approach, especially since common operating systems ignore any random bitstring when appended to an executable: such a program will run unaltered. Thus one can imagine constructing a chosen-prefix collision for two executables: a ‘good’ program file named `Word.exe` and a ‘bad’ one named `Worse.exe`. The resulting altered files, say `Word2.exe` and `Worse2.exe`, have the same MD5 hash value and are functionally equivalent to the original files. The altered ‘good’ program `Word2.exe` can then be offered to a executable signing authority (e.g. a software publisher) and receive an ’official’ MD5 based digital signature from the publisher. This signature will be equally valid for the attacker’s `Worse2.exe` which the attacker might be able to place on an appropriate download site.

This construction affects a common functionality of MD5 hashing and may pose a practical threat.

7.6.3 Nostradamus Attack

In [8] the authors present a strategy to commit to a certain hash value and afterwards construct a document, which hashes to the committed hash value, containing an arbitrary message faster than a trivial pre-image attack. The main idea is to construct a tree-structure with a root node IHV_{k+d} and 2^d end nodes $IHV_{k,j}$ where for each node $IHV_{k+i,j}$ there is a known message block $B_{k+i,j}$ resulting to its parent node. Hence, starting from any node $IHV_{k+i,j}$ there is a known suffix consisting of message blocks $B_{k+i,j}, B_{k+i+1,j'}, \ldots$ resulting in the root node IHV_{k+d}.

Starting from an arbitrary message one can brute force search for an extended message which results in some node of this tree. Further extending this message with message blocks $B_{k+i,j}$ results in the root node IHV_{k+d}. Hence, one can commit to the hash value IHV_{k+d} and afterwards construct a document containing an arbitrary message resulting in this hash value. The complexity of this attack depends on the number of nodes 2^d, constructing the tree-structure costs approx. $2^{(n+d)/2+2}$ compressions (where $n = 128$ is the bit length of the MD5 hash value) and finding the extended message resulting in some node costs approx. 2^{n-d} compressions. This attack is not practical as the total cost is at least 2^{86} compressions.

A variant of this attack is now feasible using chosen-prefix collisions. Suppose we have r messages and we want to commit to a certain hash value without committing to one of the messages specifically. Using $r-1$ chosen-prefix collisions we can construct r documents containing these r messages all with the same hash value. When committing to this hash value, afterwards we can still show any one of the r documents to achieve some malicious goal. E.g. predicting the next European Soccer Champion in a bet with large winnings.

This is the only attack we could think of where fraud cannot be revealed, as only one of the colliding messages is made public and there is no other message to hold it against to reveal fraud.

7.7 Remarks on Complexity

The amount of work required to construct a chosen-prefix collision is hard to estimate, since it is difficult to estimate the complexity for constructing the differential paths involved and finding the actual near-collision blocks. However, in our example construction of colliding certificates the work we spent in our birthday search outweighed by far the amount of work we spent in constructing the 8 differential paths and finding the actual near-collisions blocks.

Our chosen-prefix example was constructed in about 2^{52} compressions, which is much faster than the brute-force approach of about 2^{64} compressions. One can do even better using the improved birthday search, however this has not been tried yet.

8 Project HashClash using the BOINC framework

For this work we maintained the project HashClash at `http://boinc.banaan.org/hashclash/` which is a distributed computing project based on the BOINC framework. BOINC is a software platform for distributed computing using volunteer computer resources. Each project can operate completely on its own and can present work through its servers. Anybody can then use a BOINC client to register with the project. The BOINC client will then fetch, process and return workunits of the project while maintaining a background profile, i.e. as a screensaver, on the volunteer's computer. A project can customize the whole BOINC framework to its own needs, whereas the volunteer can use a standard BOINC client independent of the projects it wants to join.

A BOINC project consists of a database, data server(s), scheduling server(s), a web interface and the project backend possibly all on the same physical server as in the case of our project HashClash. The project backend is used to insert applications and workunits into the BOINC project and to receive and process returned workunits. The files of the applications and workunits are then stored on the data servers. The database contains all information about the applications, workunits, participants, participants computers and (un)returned results and is maintained by the project backend and scheduling server(s).

When a participants computer connects to the BOINC project it will use a scheduling server to request work. The scheduling server will then assign one or more workunits (if available) to the computer, after which the BOINC client will download the application and workunit files needed from the data servers. When the participants computer finishes computing a workunit it will upload all result files to the data servers and report to a scheduling server that it has finished and possibly requesting new work.

In return for the volunteered cpu-cycles the project maintains a credit system. The volunteers can compete with other users with their credit gained by donating cpu-cycles and even grouping into teams is possible. This creates a situation in which volunteers driven by competition want to donate more cpu-cycles.

Currently, there are 2752 registered volunteers most of which are part of one of the 417 teams, running a total of 8686 pc's. At its peak, the combined effort of these volunteers was about 400 Gflops. The project HashClash volunteers community was quite active and even requested for the HashClash logo competition we held (see `http://boinc.banaan.org/hashclash/logos.php`).

Using project HashClash we performed the birthday search, as shown in subsubsection 7.5.2 for our chosen-prefix collision example, by sending out workunits that generate a birthday trail starting from a given random startpoint and ending in a distinguished point. Locally, we calculated the actual collisions when two trails ending in the same distinguished point were found.

On our projects webpage we maintained a list of all found collisions and for each the two users who generated the two trails involved, whether it was useful (different IHV_3's) and how many near-collision blocks are required to eliminate the resulting δIHV_4.

In the second phase of our project we used project HashClash to distribute the work involved in finding a full differential path by connecting a lower and upper differential path given large sets of each. Using the Elegast cluster we precomputed these large sets of lower and upper differential paths and performed the collision finding when a full differential path was found.

9 Conclusion

This work presented several results related to constructing collisions for MD5. We have presented three improvements speeding up the attack by Wang et al. and also MD5 collision finding in general, namely a method to find Q_t bitconditions which satisfy T_t restrictions [21], five new differential paths to be used together with Wang's original differential paths, and our near-collision search algorithm which uses Klima's tunnels.

Together these improvements allow us to find Wang-type collisions for MD5 in approx. $2^{24.8}$ compressions or approx. 10 seconds on a 2.6Ghz Pentium4 for random IV's, here IV is the IHV used to compress to first collision block. Note that the number of compressions we show here are the work-equivalent of finding collisions instead of simply the number of different message blocks we've tried, i.e. we can find collisions on average as fast as computing approx. $2^{24.8}$ compressions. If we restrict ourselves to using recommended IV's (see subsection 5.2) and the MD5 $IV = IHV_0$ we can find collisions in even approx. $2^{24.1}$ compressions (6.2 seconds) and $2^{23.5}$ compressions (4.2 seconds), respectively. This is a large improvement over the original attack (which took approx. 2^{39} compressions using the MD5 IV) and earlier improvements where finding a single collision could take several hours on such a pc. The method of Klima [10] using tunnels is a bit slower than ours taking approx. $2^{26.3}$ compressions (28 seconds) to find collisions using the MD5 IV (the easiest case). Our earlier paper [21] (containing the improvements on satisfying T_t restrictions, our first collision finding algorithms (see Algorithms 5.1 and 5.2) and the notion of recommended IV's) was submitted to the IACR Cryptology ePrint Archive and parts of this paper were used in the book [20].

Furthermore, we presented the first automated way to construct differential paths for MD5 and showed its practicality by constructing several new differential paths (see Appendix). As mentioned above, five differential paths to speed up finding Wang-type collisions, and another eight were used in the next result.

Our most significant result is the joint work with Arjen Lenstra and Benne de Weger [22], where we have shown how to use our differential path construction method to build chosen-prefix collisions. Starting with two arbitrary different messages M, M', a chosen-prefix collision consists of these messages extended with constructed suffixes S, S' such that $MD5(M\|S) = MD5(M'\|S')$. Hence, chosen-prefix collisions allow more advanced abuse scenarios than Wang-type collisions where the only difference between colliding messages is contained in two random looking blocks. To show that chosen-prefix collisions for MD5 are feasible, we have constructed an example chosen-prefix collision consisting of two X.509 certificates with different identities but identical signatures. Our construction required substantial cpu-time, however chosen-prefix collisions might be constructed much faster by using the improved birthday search (see subsection 7.4) and allowing more near-collision blocks (about 14). Our joint work [22] was accepted at EuroCrypt 2007 and has been chosen by the program committee to be one of the three notable papers which were invited to submit their work to the Journal of Cryptology.

As part of this research we maintained the HashClash project, which is a distributed computing project using the BOINC framework. Volunteers all over the world could join our project and donate idle cpu-cycles to process computational jobs. The amount of volunteers joining our project and their enthusiasm was unexpected. Within the HashClash community we even held a logo-designing contest upon their request. It appears that the BOINC community is enthusiastic to help further such cryptography related projects, even without a good understanding of the underlying theory. With literally thousands of pc's working for our project (even if only for a small fraction of their time), we completed our chosen-prefix collisions much faster than we would have without them.

References

[1] Ivan Damgård, *A design principle for hash functions*, CRYPTO 1989 (Gilles Brassard, ed.), LNCS, vol. 435, Springer, 1989, pp. 416–427.

[2] M. Daum and S. Lucks, *Attacking hash functions by poisoned messages, the story of alice and her boss*, `http://www.cits.rub.de/MD5Collisions/`.

[3] Bert den Boer and Antoon Bosselaers, *Collisions for the compression function of MD5*, EUROCRYPT 1993 (Tor Helleseth, ed.), LNCS, vol. 765, Springer, 1993, pp. 293–304.

[4] Hans Dobbertin, *Cryptanalysis of MD5 compress*, 1996, presented at the rump session of Eurocrypt'96.

[5] M. Gebhardt, G. Illies, and W. Schindler, *Note on practical value of single hash collisions for special file formats*, NIST First Cryptographic Hash Workshop, 2005, `http://csrc.nist.gov/pki/HashWorkshop/2005/Oct31_Presentations/Illies_NIST_05.pdf`.

[6] Philip Hawkes, Michael Paddon, and Gregory G. Rose, *Musings on the Wang et al. MD5 collision*, Cryptology ePrint Archive, Report 2004/264, 2004, `http://eprint.iacr.org/2004/264`.

[7] D. Kaminsky, *MD5 to be considered harmful someday*, Cryptology ePrint Archive, Report 2004/357, 2004, `http://eprint.iacr.org/2004/357`.

[8] John Kelsey and Tadayoshi Kohno, *Herding hash functions and the nostradamus attack*, EUROCRYPT 2006 (Serge Vaudenay, ed.), LNCS, vol. 4004, Springer, 2006, pp. 183–200.

[9] Vlastimil Klima, *Finding MD5 collisions on a notebook PC using multi-message modifications*, Cryptology ePrint Archive, Report 2005/102, 2005, `http://eprint.iacr.org/2005/102`.

[10] Vlastimil Klima, *Tunnels in hash functions: MD5 collisions within a minute*, Cryptology ePrint Archive, Report 2006/105, 2006, `http://eprint.iacr.org/2006/105`.

[11] Arjen K. Lenstra and Benne de Weger, *On the possibility of constructing meaningful hash collisions for public keys*, ACISP 2005 (Colin Boyd and Juan Manuel González Nieto, eds.), LNCS, vol. 3574, Springer, 2005, pp. 267–279.

[12] Jie Liang and Xuejia Lai, *Improved collision attack on hash function MD5*, Cryptology ePrint Archive, Report 2005/425, 2005, `http://eprint.iacr.org/2005/425`.

[13] Ralph C. Merkle, *One way hash functions and DES*, CRYPTO 1989 (Gilles Brassard, ed.), LNCS, vol. 435, Springer, 1989, pp. 428–446.

[14] Ondrej Mikle, *Practical attacks on digital signatures using MD5 message digest*, Cryptology ePrint Archive, Report 2004/356, 2004, `http://eprint.iacr.org/2004/356`.

[15] Jamer A. Muir and Douglas R. Stinson, *Minimality and other properties of the width-w nonadjacent form*, Mathematics of Computation, vol. 75, 2006, pp. 369–384.

[16] National Information Standards Organisation, *FIPS PUB 180-1: Secure hash standard*, April 1995, `http://www.itl.nist.gov/fipspubs/fip180.htm`.

[17] R.L. Rivest, *The MD5 Message-Digest algorithm*, Internet RFC, April 1992, RFC 1321.

[18] Yu Sasaki, Yusuke Naito, Noboru Kunihiro, and Kazuo Ohta, *Improved collision attack on MD5*, Cryptology ePrint Archive, Report 2005/400, 2005, `http://eprint.iacr.org/2005/400`.

[19] Yu Sasaki, Yusuke Naito, Jun Yajima, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta, *How to construct sufficient condition in searching collisions of MD5*, Cryptology ePrint Archive, Report 2006/074, 2006, `http://eprint.iacr.org/2006/074`.

[20] Mark Stamp and Richard M. Low, *Applied cryptanalysis*, Wiley, 2007.

[21] Marc Stevens, *Fast collision attack on MD5*, Cryptology ePrint Archive, Report 2006/104, 2006, `http://eprint.iacr.org/2006/104`.

[22] Marc Stevens, Arjen Lenstra, and Benne de Weger, *Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities*, EUROCRYPT 2007 (Moni Naor, ed.), LNCS, vol. 4515, Springer, 2007, pp. 1–22.

[23] Paul C. van Oorschot and Michael J. Wiener, *Parallel collision search with cryptanalytic applications*, Journal of Cryptology **12** (1999), no. 1, 1–28.

[24] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu, *Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD*, Cryptology ePrint Archive, Report 2004/199, 2004, `http://eprint.iacr.org/2004/199`.

[25] Xiaoyun Wang and Hongbo Yu, *How to break MD5 and other hash functions*, EUROCRYPT 2005 (Ronald Cramer, ed.), LNCS, vol. 3494, Springer, 2005, pp. 19–35.

[26] Jun Yajima and Takeshi Shimoyama, *Wang's sufficient conditions of MD5 are not sufficient*, Cryptology ePrint Archive, Report 2005/263, 2005, `http://eprint.iacr.org/2005/263`.

A MD5 Constants and Message Block Expansion

Table A-1: MD5 Addition and Rotation Constants and message block expansion.

t	AC_t	RC_t	W_t
0	$\texttt{d76aa478}_{16}$	7	m_0
1	$\texttt{e8c7b756}_{16}$	12	m_1
2	$\texttt{242070db}_{16}$	17	m_2
3	$\texttt{c1bdceee}_{16}$	22	m_3
4	$\texttt{f57c0faf}_{16}$	7	m_4
5	$\texttt{4787c62a}_{16}$	12	m_5
6	$\texttt{a8304613}_{16}$	17	m_6
7	$\texttt{fd469501}_{16}$	22	m_7
8	$\texttt{698098d8}_{16}$	7	m_8
9	$\texttt{8b44f7af}_{16}$	12	m_9
10	$\texttt{ffff5bb1}_{16}$	17	m_{10}
11	$\texttt{895cd7be}_{16}$	22	m_{11}
12	$\texttt{6b901122}_{16}$	7	m_{12}
13	$\texttt{fd987193}_{16}$	12	m_{13}
14	$\texttt{a679438e}_{16}$	17	m_{14}
15	$\texttt{49b40821}_{16}$	22	m_{15}

t	AC_t	RC_t	W_t
16	$\texttt{f61e2562}_{16}$	5	m_1
17	$\texttt{c040b340}_{16}$	9	m_6
18	$\texttt{265e5a51}_{16}$	14	m_{11}
19	$\texttt{e9b6c7aa}_{16}$	20	m_0
20	$\texttt{d62f105d}_{16}$	5	m_5
21	$\texttt{02441453}_{16}$	9	m_{10}
22	$\texttt{d8a1e681}_{16}$	14	m_{15}
23	$\texttt{e7d3fbc8}_{16}$	20	m_4
24	$\texttt{21e1cde6}_{16}$	5	m_9
25	$\texttt{c33707d6}_{16}$	9	m_{14}
26	$\texttt{f4d50d87}_{16}$	14	m_3
27	$\texttt{455a14ed}_{16}$	20	m_8
28	$\texttt{a9e3e905}_{16}$	5	m_{13}
29	$\texttt{fcefa3f8}_{16}$	9	m_2
30	$\texttt{676f02d9}_{16}$	14	m_7
31	$\texttt{8d2a4c8a}_{16}$	20	m_{12}

t	AC_t	RC_t	W_t
32	$\texttt{fffa3942}_{16}$	4	m_5
33	$\texttt{8771f681}_{16}$	11	m_8
34	$\texttt{6d9d6122}_{16}$	16	m_{11}
35	$\texttt{fde5380c}_{16}$	23	m_{14}
36	$\texttt{a4beea44}_{16}$	4	m_1
37	$\texttt{4bdecfa9}_{16}$	11	m_4
38	$\texttt{f6bb4b60}_{16}$	16	m_7
39	$\texttt{bebfbc70}_{16}$	23	m_{10}
40	$\texttt{289b7ec6}_{16}$	4	m_{13}
41	$\texttt{eaa127fa}_{16}$	11	m_0
42	$\texttt{d4ef3085}_{16}$	16	m_3
43	$\texttt{04881d05}_{16}$	23	m_6
44	$\texttt{d9d4d039}_{16}$	4	m_9
45	$\texttt{e6db99e5}_{16}$	11	m_{12}
46	$\texttt{1fa27cf8}_{16}$	16	m_{15}
47	$\texttt{c4ac5665}_{16}$	23	m_2

t	AC_t	RC_t	W_t
48	$\texttt{f4292244}_{16}$	6	m_0
49	$\texttt{432aff97}_{16}$	10	m_7
50	$\texttt{ab9423a7}_{16}$	15	m_{14}
51	$\texttt{fc93a039}_{16}$	21	m_5
52	$\texttt{655b59c3}_{16}$	6	m_{12}
53	$\texttt{8f0ccc92}_{16}$	10	m_3
54	$\texttt{ffeff47d}_{16}$	15	m_{10}
55	$\texttt{85845dd1}_{16}$	21	m_1
56	$\texttt{6fa87e4f}_{16}$	6	m_8
57	$\texttt{fe2ce6e0}_{16}$	10	m_{15}
58	$\texttt{a3014314}_{16}$	15	m_6
59	$\texttt{4e0811a1}_{16}$	21	m_{13}
60	$\texttt{f7537e82}_{16}$	6	m_4
61	$\texttt{bd3af235}_{16}$	10	m_{11}
62	$\texttt{2ad7d2bb}_{16}$	15	m_2
63	$\texttt{eb86d391}_{16}$	21	m_9

B Differential Paths for Two Block Collisions

B.1 Wang et al.'s Differential Paths

Table B-1: Wang et al.'s first block differential

$$\delta m_4 = +2^{31}, \quad \delta m_{11} = +2^{15}, \quad \delta m_{14} = +2^{31}, \quad \delta m_i = 0, i \notin \{4, 11, 14\}$$

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
$0-3$	$-$	$-$	$-$	$-$	$\cdot$
4	$-$	$-$	2^{31}	2^{31}	7
5	$+2^6 \ldots +2^{21}, -2^{22}$	$2^{11}+2^{19}$	$-$	$2^{11}+2^{19}$	12
6	$-2^6+2^{23}+2^{31}$	$-2^{10}-2^{14}$	$-$	$-2^{10}-2^{14}$	17
7	$+2^0 \ldots +2^4, -2^5, +2^6 \ldots +2^{10}$ $-2^{11}, -2^{23} \ldots -2^{25}, +2^{26} \ldots +2^{31}$	$-2^2+2^5+2^{10}$ $+2^{16}-2^{25}-2^{27}$	$-$	$-2^2+2^5+2^{10}$ $+2^{16}-2^{25}-2^{27}$	22
8	$+2^0+2^{15}-2^{16}+2^{17}$ $+2^{18}+2^{19}-2^{20}-2^{23}$	$2^6+2^8+2^{10}$ $+2^{16}-2^{24}+2^{31}$	$-$	$2^8+2^{10}+2^{16}$ $-2^{24}+2^{31}$	7
9	$-2^0+2^1+2^6+2^7-2^8-2^{31}$	$2^0+2^6-2^{20}$ $-2^{23}+2^{26}+2^{31}$	$-$	$2^0-2^{20}+2^{26}$	12
10	$-2^{12}+2^{13}+2^{31}$	$2^0+2^6+2^{13}-2^{23}$	$-$	$2^{13}-2^{27}$	17
11	$+2^{30}+2^{31}$	-2^0-2^8	2^{15}	$-2^8-2^{17}-2^{23}$	22
12	$+2^7-2^8, +2^{13} \ldots +2^{18}, -2^{19}+2^{31}$	$2^7+2^{17}+2^{31}$	$-$	$2^0+2^6+2^{17}$	7
13	$-2^{24}+2^{25}+2^{31}$	$-2^{13}+2^{31}$	$-$	-2^{12}	12
14	$+2^{31}$	$2^{18}+2^{31}$	2^{31}	$2^{18}-2^{30}$	17
15	$+2^3-2^{15}+2^{31}$	$2^{25}+2^{31}$	$-$	$-2^7-2^{13}+2^{25}$	22
16	$-2^{29}+2^{31}$	2^{31}	$-$	2^{24}	5
17	$+2^{31}$	2^{31}	$-$	$-$	9
18	$+2^{31}$	2^{31}	2^{15}	2^3	14
19	$+2^{17}+2^{31}$	2^{31}	$-$	-2^{29}	20
20	$+2^{31}$	2^{31}	$-$	$-$	5
21	$+2^{31}$	2^{31}	$-$	$-$	9
22	$+2^{31}$	2^{31}	$-$	2^{17}	14
23	$-$	$-$	2^{31}	$-$	20
24	$-$	2^{31}	$-$	$-$	5
25	$-$	$-$	2^{31}	$-$	9
$26-33$	$-$	$-$	$-$	$-$	$\cdot$
34	$-$	$-$	2^{15}	2^{15}	16
35	$\delta Q_{35} = 2^{31}$	2^{31}	2^{31}	$-$	23
36	$\delta Q_{36} = 2^{31}$	$-$	$-$	$-$	4
37	$\delta Q_{37} = 2^{31}$	2^{31}	2^{31}	$-$	11
$38-49$	$\delta Q_t = 2^{31}$	2^{31}	$-$	$-$	$\cdot$
50	$\delta Q_{50} = 2^{31}$	$-$	2^{31}	$-$	15
$51-59$	$\delta Q_t = 2^{31}$	2^{31}	$-$	$-$	$\cdot$
60	$\delta Q_{60} = 2^{31}$	$-$	2^{31}	$-$	6
61	$\delta Q_{61} = 2^{31}$	2^{31}	2^{15}	2^{15}	10
62	$\delta Q_{62} = 2^{31}+2^{25}$	2^{31}	$-$	$-$	15
63	$\delta Q_{63} = 2^{31}+2^{25}$	2^{31}	$-$	$-$	21
64	$\delta Q_{64} = 2^{31}+2^{25}$	$\times$	$\times$	$\times$	$\times$

Table B-2: Wang et al.'s second block differential

$$\delta m_4 = -2^{31}, \quad \delta m_{11} = -2^{15}, \quad \delta m_{14} = -2^{31}, \quad \delta m_i = 0, i \notin \{4, 11, 14\}$$

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
-3	$+2^{31}$	$\times$	$\times$	$\times$	$\times$
-2	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
-1	$-2^{25}+2^{26}+2^{31}$	$\times$	$\times$	$\times$	$\times$
0	$+2^{25}+2^{31}$	2^{31}	$-$	$-$	7
1	$+2^{25}+2^{31}$	2^{31}	$-$	2^{25}	12
2	$+2^{5}+2^{25}+2^{31}$	2^{25}	$-$	$2^{31}+2^{26}$	17
3	$-2^{5}-2^{6}+2^{7}-2^{11}+2^{12}$ $-2^{16}\ldots-2^{20},+2^{21}$ $-2^{25}\ldots-2^{29},+2^{30}+2^{31}$	$-2^{11}-2^{21}+2^{25}$ $-2^{27}+2^{31}$	$-$	$-2^{11}-2^{21}-2^{26}$	22
4	$+2^{1}+2^{2}+2^{3}-2^{4}+2^{5}$ $-2^{25}+2^{26}+2^{31}$	$2^{1}-2^{3}-2^{18}$ $+2^{26}+2^{30}$	2^{31}	$2^{1}+2^{2}-2^{18}$ $+2^{25}+2^{26}+2^{30}$	7
5	$+2^{0}-2^{6}+2^{7}+2^{8}-2^{9}$ $-2^{10}-2^{11}+2^{12}+2^{31}$	$-2^{4}-2^{5}-2^{8}-2^{20}$ $-2^{25}-2^{26}+2^{28}+2^{30}$	$-$	$-2^{4}-2^{8}-2^{20}$ $-2^{26}+2^{28}-2^{30}$	12
6	$+2^{16}-2^{17}+2^{20}-2^{21}+2^{31}$	$2^{3}-2^{5}-2^{10}-2^{11}$ $-2^{16}-2^{21}-2^{25}$	$-$	$2^{3}-2^{10}-2^{21}-2^{31}$	17
7	$+2^{6}+2^{7}+2^{8}-2^{9}$ $+2^{27}-2^{28}+2^{31}$	$2^{16}-2^{27}+2^{31}$	$-$	$-2^{1}+2^{5}+2^{16}$ $+2^{25}-2^{27}$	22
8	$-2^{15}+2^{16}-2^{17}+2^{23}$ $+2^{24}+2^{25}-2^{26}+2^{31}$	$-2^{6}+2^{16}+2^{25}$	$-$	$2^{0}+2^{8}+2^{9}$ $+2^{16}+2^{25}-2^{31}$	7
9	$-2^{0}+2^{1},-2^{6}\ldots-2^{8},+2^{9}+2^{31}$	$2^{0}+2^{16}-2^{26}+2^{31}$	$-$	$2^{0}-2^{20}-2^{26}$	12
10	$+2^{12}+2^{31}$	$2^{6}+2^{31}$	$-$	-2^{27}	17
11	$+2^{31}$	2^{31}	-2^{15}	$-2^{17}-2^{23}$	22
12	$-2^{7},+2^{13}\ldots+2^{18}-2^{19}+2^{31}$	$2^{17}+2^{31}$	$-$	$2^{0}+2^{6}+2^{17}$	7
13	$-2^{24}\ldots-2^{29},+2^{30}+2^{31}$	$-2^{13}+2^{31}$	$-$	-2^{12}	12
14	$+2^{31}$	$2^{18}+2^{30}$	2^{31}	$2^{18}+2^{30}$	17
15	$+2^{3}+2^{15}+2^{31}$	$-2^{25}+2^{31}$	$-$	$-2^{7}-2^{13}-2^{25}$	22
16	$-2^{29}+2^{31}$	2^{31}	$-$	2^{24}	5
17	$+2^{31}$	2^{31}	$-$	$-$	9
18	$+2^{31}$	2^{31}	-2^{15}	2^{3}	14
19	$+2^{17}+2^{31}$	2^{31}	$-$	-2^{29}	20
20	$+2^{31}$	2^{31}	$-$	$-$	5
21	$+2^{31}$	2^{31}	$-$	$-$	9
22	$+2^{31}$	2^{31}	$-$	2^{17}	14
23	$-$	$-$	2^{31}	$-$	20
24	$-$	2^{31}	$-$	$-$	5
25	$-$	$-$	2^{31}	$-$	9
$26-33$	$-$	$-$	$-$	$-$	$\cdot$
34	$-$	$-$	-2^{15}	-2^{15}	16
35	$\delta Q_{35}=2^{31}$	2^{31}	2^{31}	$-$	23
36	$\delta Q_{36}=2^{31}$	$-$	$-$	$-$	4
37	$\delta Q_{37}=2^{31}$	2^{31}	2^{31}	$-$	11
$38-49$	$\delta Q_t=2^{31}$	2^{31}	$-$	$-$	$\cdot$
50	$\delta Q_{50}=2^{31}$	$-$	2^{31}	$-$	15
$51-59$	$\delta Q_t=2^{31}$	2^{31}	$-$	$-$	$\cdot$
60	$\delta Q_{60}=2^{31}$	$-$	2^{31}	$-$	6
61	$\delta Q_{61}=2^{31}$	2^{31}	-2^{15}	-2^{15}	10
62	$\delta Q_{62}=2^{31}-2^{25}$	2^{31}	$-$	$-$	15
63	$\delta Q_{63}=2^{31}-2^{25}$	2^{31}	$-$	$-$	21
64	$\delta Q_{64}=2^{31}-2^{25}$	$\times$	$\times$	$\times$	$\times$

B.2 Modified Sufficient Conditions for Wang's Differential Paths

Table B-3: Modified first block bitconditions

t	Conditions on Q_t: $b_{31} \dots b_0$	#
3	`........ ....0... ....0... .0......`	3
4	`1....... 0^^^1^^^ ^^^^1^^^ ^011....`	$19+2$
5	`1000100. 01..0000 00000000 0010.1.1`	$22+5$
6	`0000001^ 01111111 10111100 0100^0^1`	32
7	`00000011 11111110 11111000 00100000`	32
8	`00000001 1..10001 0.0.0101 01000000`	28
9	`11111011 ...10000 0.1^1111 00111101`	28
10	`0111.... 0..11111 1101...0 01....00`	$17+2$
11	`0010.... ....0001 1100...0 11....10`	$15+2$
12	`000...^^ ....1000 0001...1 0.......`	$14+1$
13	`01....01 ....1111 111....0 0...1...`	14
14	`0.0...00 ....1011 111....1 1...1...`	14
15	`0.1...01 .......0 1....... ....0...`	$6+1$
16	`0!1..... ......!. ........ ........`	$2+2$
17	`0!...... ......0. ^....... ....^...`	$4+1$
18	`0.^..... ......1. ........ ........`	3
19	`0....... ......0. ........ ........`	2
20	`0....... .....!.. ........ ........`	$1+1$
21	`0....... ......^. ........ ........`	2
22	`0....... ........ ........ ........`	1
23	`0....... ........ ........ ........`	1
24	`1....... ........ ........ ........`	1
	Sub-total # conditions	278
$25-45$	`........ ........ ........ ........`	0
46	`I....... ........ ........ ........`	0
47	`J....... ........ ........ ........`	0
48	`I....... ........ ........ ........`	1
49	`J....... ........ ........ ........`	1
50	`K....... ........ ........ ........`	1
51	`J....... ........ ........ ........`	1
52	`K....... ........ ........ ........`	1
53	`J....... ........ ........ ........`	1
54	`K....... ........ ........ ........`	1
55	`J....... ........ ........ ........`	1
56	`K....... ........ ........ ........`	1
57	`J....... ........ ........ ........`	1
58	`K....... ........ ........ ........`	1
59	`J....... ........ ........ ........`	1
60	`I....... ........ ........ ........`	1
61	`J....... ........ ........ ........`	1
62	`I....... ........ ........ ........`	1
63	`J....... ........ ........ ........`	1
64	`........ ........ ........ ........`	0
	Sub-total # conditions	16
	Sub-total # IV conditions from 2nd block	8
	Total # conditions	302

Note: $I, J, K \in \{0, 1\}$ and $K = \overline{I}$.

Table B-4: Modified second block bitconditions

t	Conditions on Q_t: $b_{31} \dots b_0$	#
-2	A.....0.	(1)
-1	A....01.	(3)
0	A....00. ....0.....	(4)
	Total # IV conditions for 1st block	(8)
1	B...010. ..1....0 ....0... .10.....	$8+1$
2	B^^^110. ..0^^^^0 1..^1... ^10..00.	$20+1$
3	B011111. ..011111 0..01..1 011^^11.	$23+1$
4	B011101. ..000100 ...00^^0 0001000^	26
5	A10010.. ..101111 ...01110 01010000	25
6	A..0010. 1.10..10 11.01100 01010110	$24+1$
7	B..1011^ 1.00..01 10.11110 00.....1	$20+1$
8	B..00100 0.11..10 1.....11 111...^0	$18+1$
9	B..11100 0.....01 0..^..01 110...01	$17+1$
10	B....111 1....011 11001.11 11....00	$18+1$
11	B....... ....^101 11000.11 11....11	$15+1$
12	B^^^^^^^ ....1000 0001.... 1.......	17
13	A0111111 0...1111 111..... 0...1...	$17+1$
14	A1000000 1...1011 111..... 1...1...	$17+1$
15	01111101 .... 00...... ....0...	$10+1$
16	0.10.... ......!.	$2+2$
17	0!...... ......0. ^....... ....^...	$4+1$
18	0.^..... ......1.	3
19	0....... ......0.	2
20	0....... .....!..	1
21	0....... ......^.	2
22	0.......	1
23	0.......	1
24	1.......	1
	Sub-total # conditions	307
$25-45$	....	
46	I.......	0
47	J.......	0
48	I.......	1
49	J.......	1
50	K.......	1
51	J.......	1
52	K.......	1
53	J.......	1
54	K.......	1
55	J.......	1
56	K.......	1
57	J.......	1
58	K.......	1
59	J.......	1
60	I.......	1
61	J.......	1
62	I.......	1
63	J.......	1
64	....	0
	Sub-total # conditions	16
	Total # conditions	323

Note: $A, B, I, J, K \in \{0, 1\}$ and $B = \overline{A}$, $K = \overline{I}$.

B.3 New First Block Differential Path

Table B-5: New first block differential path

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
$0-3$	$-$	$-$	$-$	$-$	$\cdot$
4	$-$	$-$	2^{31}	2^{31}	7
5	$-2^{6}\ldots-2^{24}+2^{25}$	$-2^{8}+2^{14}-2^{19}$ $-2^{23}+2^{25}$	$-$	$-2^{8}+2^{14}-2^{19}$ $-2^{23}+2^{25}$	12
6	$+2^{0}-2^{1}+2^{3}-2^{4}$ $+2^{5}-2^{6}-2^{7}+2^{8}+2^{20}$ $+2^{21}-2^{22}+2^{26}-2^{31}$	$2^{3}-2^{9}+2^{15}$ $+2^{18}-2^{20}-2^{22}$	$-$	$2^{3}-2^{9}+2^{15}$ $+2^{18}-2^{20}-2^{22}$	17
7	$-2^{6}+2^{31}$	$-2^{0}+2^{6}-2^{10}$ $+2^{13}-2^{25}$	$-$	$-2^{0}+2^{6}-2^{10}$ $+2^{13}-2^{25}$	22
8	$-2^{0}+2^{3}-2^{6}-2^{15}$ $-2^{22}+2^{28}+2^{31}$	$-2^{5}+2^{8}+2^{15}$ $-2^{21}+2^{26}-2^{28}$	$-$	$2^{5}+2^{8}+2^{15}$ $-2^{21}+2^{26}-2^{28}$	7
9	$+2^{0}-2^{6}+2^{12}+2^{31}$	$-2^{0}+2^{3}-2^{6}+2^{31}$	$-$	$-2^{1}+2^{5}-2^{20}+2^{26}$	12
10	$-2^{12}+2^{17}+2^{31}$	$2^{0}-2^{6}+2^{12}+2^{31}$	$-$	$2^{0}-2^{7}+2^{12}$	17
11	$-2^{12}+2^{18}-2^{24}$ $+2^{29}+2^{31}$	$2^{0}-2^{6}-2^{17}$ $-2^{29}+2^{31}$	2^{15}	$2^{3}-2^{7}-2^{17}$ $-2^{22}-2^{28}$	22
12	$-2^{7}-2^{13}+2^{24}+2^{31}$	$2^{7}-2^{12}+2^{31}$	$-$	$2^{0}+2^{6}$	7
13	$+2^{24}+2^{31}$	2^{31}	$-$	$-2^{12}+2^{17}$	12
14	$+2^{29}+2^{31}$	$2^{24}+2^{29}+2^{31}$	2^{31}	$-2^{12}+2^{18}-2^{30}$	17
15	$+2^{3}-2^{15}-2^{31}$	$2^{24}+2^{31}$	$-$	$-2^{7}-2^{13}+2^{25}$	22
16	$-2^{29}-2^{31}$	2^{31}	$-$	2^{24}	5
17	-2^{31}	$-2^{29}+2^{31}$	$-$	$-$	9
18	-2^{31}	2^{31}	2^{15}	2^{3}	14
19	$+2^{17}-2^{31}$	2^{31}	$-$	-2^{29}	20
20	-2^{31}	2^{31}	$-$	$-$	5
21	-2^{31}	2^{31}	$-$	$-$	9
22	-2^{31}	2^{31}	$-$	2^{17}	14
23	$-$	$-$	2^{31}	$-$	20
24	$-$	2^{31}	$-$	$-$	5
25	$-$	$-$	2^{31}	$-$	9

Table B-6: New first block conditions

t	Conditions on Q_t: $b_{31} \dots b_0$	#
3	`.......1 .1111... .1....01 11......`	10
4	`0.....^0 ^0000^^^ ^0^^^^10 11..0...`	22
5	`01...^01 11111111 11111111 11^^1.^^`	28
6	`10.1.000 01001011 10000010 11010.10`	29
7	`0..0.010 01000000 00011011 .1000.11`	27
8	`0!.0.0.. .101.... 1..1...0 11010.11`	17
9	`0!10...0 .0...1^. 0..0.... 011.1..0`	15
10	`0.01...0 .1...00. 1..1.... 1...1..1`	12
11	`0!0....1 .....01. ..^1.... 00.....0`	11
12	`0!0....0 ..!..01. ..1..... 1.......`	9
13	`0.1....0 .....1.. 1.01.... 0...1...`	9
14	`0!0..... ........ 1.1..... 1...1...`	7
15	`1.0....0 .......! 1....... ....0...`	6
16	`1!1..... ......!. ........ ........`	4
17	`1!...... ......0. ^....... ....^...`	5
18	`1.^..... ......1. ........ ........`	3
19	`1....... ......0. ........ ........`	2
20	`1....... .....!.. ........ ........`	2
21	`1....... ......^. ........ ........`	2
22	`1....... ........ ........ ........`	1
23	`0....... ........ ........ ........`	1
24	`1....... ........ ........ ........`	1
	Subtotal # conditions	223

B.4 New Second Block Differential Paths

B.4.1 New Second Block Differential Path nr. 1

Table B-7: New second block differential path nr. 1

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
-3	$+2^{31}$	$\times$	$\times$	$\times$	$\times$
-2	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
-1	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
0	$+2^{25}+2^{31}$	$2^{25}+2^{31}$	–	2^{25}	7
1	$+2^{0}+2^{25}+2^{31}$	$2^{25}+2^{31}$	–	2^{26}	12
2	$+2^{0}+2^{6}+2^{25}+2^{31}$	$2^{25}+2^{31}$	–	2^{26}	17
3	$+2^{0}+2^{6}+2^{11}+2^{25}+2^{31}$	$2^{0}-2^{11}+2^{25}+2^{31}$	–	$2^{0}-2^{11}+2^{26}$	22
4	$+2^{0}-2^{1}-2^{6}-2^{7}$ $+2^{8}+2^{11}+2^{16}+2^{22}$ $-2^{25}-2^{26}+2^{27}+2^{31}$	$2^{0}-2^{6}+2^{8}+2^{11}$ $-2^{16}-2^{22}+2^{25}$ $-2^{27}+2^{31}$	2^{31}	$2^{1}-2^{6}+2^{8}$ $+2^{11}-2^{16}-2^{22}$ $-2^{26}+2^{31}$	7
5	$+2^{0}+2^{2}+2^{3}+2^{4}-2^{5}+2^{8}$ $+2^{11}-2^{13}-2^{15}-2^{17}+2^{19}$ $+2^{22}+2^{23}+2^{24}+2^{29}+2^{30}$	$2^{0}+2^{3}-2^{5}+2^{7}$ $+2^{11}+2^{16}+2^{19}$ $+2^{22}-2^{25}-2^{29}$	–	$2^{1}+2^{3}+2^{5}+2^{7}$ $+2^{11}+2^{16}+2^{19}$ $+2^{22}-2^{29}+2^{31}$	12
6	$+2^{1}+2^{8}\ldots+2^{17}-2^{18}$ $-2^{20}+2^{21}-2^{22}-2^{23}-2^{24}$ $+2^{26}+2^{28}+2^{29}+2^{30}+2^{31}$	$-2^{0}-2^{3}+2^{5}+2^{7}+2^{9}$ $+2^{11}-2^{13}+2^{15}-2^{17}-2^{21}$ $+2^{23}-2^{25}+2^{28}+2^{31}$	–	$-2^{3}-2^{5}-2^{8}+2^{10}$ $-2^{12}+2^{15}-2^{17}$ $-2^{21}+2^{23}+2^{28}$	17
7	$-2^{0}-2^{6}+2^{13}-2^{27}-2^{29}$	$-2^{0}+2^{3}+2^{5}+2^{7}-2^{10}$ $-2^{13}+2^{18}+2^{22}-2^{24}$ $+2^{27}-2^{29}+2^{31}$	–	$-2^{1}+2^{3}-2^{5}+2^{8}+2^{10}$ $-2^{13}+2^{16}+2^{18}-2^{23}$ $+2^{25}+2^{27}-2^{29}$	22
8	$-2^{3}+2^{8}+2^{15}+2^{17}$ $-2^{19}-2^{23}+2^{25}+2^{28}$	$2^{0}+2^{9}+2^{13}-2^{15}-2^{19}$ $+2^{21}+2^{26}-2^{28}+2^{30}$	–	$-2^{1}-2^{8}-2^{10}+2^{12}+2^{16}$ $-2^{18}-2^{21}-2^{25}$ $-2^{27}+2^{29}+2^{31}$	7
9	$-2^{0}+2^{2}-2^{6}$	$-2^{0}+2^{8}-2^{23}+2^{25}+2^{28}$	–	$2^{0}+2^{20}-2^{22}+2^{26}$	12
10	$+2^{12}$	$-2^{0}-2^{6}+2^{8}$	–	$-2^{1}+2^{7}+2^{13}-2^{27}-2^{29}$	17
11	$-2^{14}-2^{18}+2^{24}+2^{30}$	–	-2^{15}	$-2^{3}+2^{8}+2^{17}-2^{19}$ $-2^{23}+2^{25}+2^{28}$	22
12	$+2^{7}-2^{9}+2^{13}-2^{24}-2^{31}$	–	–	$-2^{0}+2^{2}-2^{6}$	7
13	$-2^{24}-2^{31}$	–	–	2^{12}	12
14	-2^{31}	$-2^{24}+2^{31}$	2^{31}	$-2^{14}-2^{18}+2^{30}$	17
15	$-2^{3}+2^{15}$	$-2^{24}+2^{31}$	–	$2^{7}-2^{9}+2^{13}-2^{25}$	22
16	$+2^{29}-2^{31}$	2^{31}	–	-2^{24}	5
17	-2^{31}	2^{31}	–	–	9
18	$+2^{31}$	–	-2^{15}	-2^{3}	14
19	$-2^{17}+2^{31}$	2^{31}	–	2^{29}	20
20	$+2^{31}$	2^{31}	–	–	5
21	$+2^{31}$	2^{31}	–	–	9
22	$+2^{31}$	2^{31}	–	-2^{17}	14
23	–	–	2^{31}	–	20
24	–	2^{31}	–	–	5
25	–	–	2^{31}	–	9

Table B-8: New second block conditions nr. 1

t	Conditions on Q_t: $b_{31} \dots b_0$	#
-2	`......0. ........ ........ ........`	(1)
-1	`^.....0. ........ ........ .......1`	(3)
0	`^.....0. ........ ........ .1.....1`	(4)
	Total # IV conditions for 1st block	(8)
1	`^.....0. ........ ....1... .1!....0`	6
2	`^...1.0. .1.....1 ....0..0 00.....0`	10
3	`^00.0^00 00..0.10 1.1.0..1 101.0.^0`	22
4	`01100110 000.1.10 1.1.0.00 110^1^10`	27
5	`.0010100 001^0^11 1^1^0^10 00100000`	31
6	`^0001001 11010100 00000000 01011000`	32
7	`0.111001 01001011 1101.100 .1011011`	29
8	`10100001 11011000 01.11100 .00.1001`	29
9	`.1111.10 1...0.0. 0.11...1 .11.00.1`	18
10	`1111..10 1...1^1. 1^.0...0 .1..10.1`	18
11	`100....0 .....1.! .1^0..^. ^1...1.1`	14
12	`.01....1 ..!..0.. .001..1. 0.......`	10
13	`^1.....1 .....1.. 110...0. 0...1...`	10
14	`100..... ........ 1.1...1. 1...1...`	8
15	`001....0 .......! 0....... ....1...`	7
16	`1!0..... ......!. ........ ........`	4
17	`1!...... ......0. ^....... ....^...`	5
18	`0.^..... ......1. ........ ........`	3
19	`0....... ......1. ........ ........`	2
20	`0....... .....!.. ........ ........`	2
21	`0....... ......^. ........ ........`	2
22	`0....... ........ ........ ........`	1
23	`0....... ........ ........ ........`	1
24	`1....... ........ ........ ........`	1
	Subtotal # conditions	292

B.4.2 New Second Block Differential Path nr. 2

Table B-9: New second block differential path nr. 2

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
-3	$+2^{31}$	$\times$	$\times$	$\times$	$\times$
-2	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
-1	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
0	$+2^{25}+2^{31}$	$2^{25}+2^{31}$	$-$	2^{25}	7
1	$+2^{0}+2^{25}+2^{31}$	$2^{25}+2^{31}$	$-$	2^{26}	12
2	$+2^{0}+2^{6}+2^{25}+2^{31}$	$2^{0}+2^{25}+2^{31}$	$-$	$2^{0}+2^{26}$	17
3	$+2^{0}+2^{6}+2^{11}$ $+2^{17}+2^{25}+2^{31}$	$2^{0}+2^{25}+2^{31}$	$-$	$2^{0}+2^{26}$	22
4	$-2^{0}-2^{1}+2^{2}-2^{6}-2^{7}+2^{8}$ $-2^{11}-2^{12}+2^{13}-2^{16}-2^{18}$ $+2^{19}+2^{22}-2^{25}+2^{26}+2^{31}$	$2^{0}+2^{2}+2^{6}+2^{8}$ $-2^{11}+2^{16}-2^{18}-2^{22}$ $-2^{25}+2^{27}+2^{31}$	2^{31}	$-2^{1}+2^{3}+2^{6}+2^{8}$ $-2^{11}+2^{16}-2^{18}$ $-2^{22}+2^{27}+2^{31}$	7
5	$+2^{0}-2^{2}-2^{3}-2^{4}+2^{5}$ $-2^{8}-2^{10}-2^{12}+2^{14}-2^{15}$ $+2^{22}-2^{23}+2^{24}+2^{29}+2^{30}$	$2^{2}-2^{6}+2^{8}-2^{11}$ $-2^{14}+2^{16}+2^{19}$ $-2^{22}-2^{25}-2^{30}$	$-$	$2^{0}+2^{2}+2^{8}-2^{11}$ $-2^{14}+2^{16}+2^{19}$ $-2^{22}+2^{30}$	12
6	$-2^{1}-2^{2}+2^{3},+2^{8}\dots+2^{15}$ $-2^{16}-2^{20}+2^{21}+2^{22}+2^{26}$ $+2^{27}+2^{29}+2^{30}-2^{31}$	$-2^{0}-2^{3}+2^{5}-2^{7}-2^{9}$ $+2^{11}-2^{13}+2^{15}+2^{17}$ $-2^{19}-2^{22}-2^{24}+2^{31}$	$-$	$-2^{3}-2^{5}-2^{9}-2^{12}$ $+2^{15}-2^{18}-2^{22}+2^{24}$	17
7	$-2^{0}-2^{2}+2^{7}+2^{27}-2^{30}$	$2^{0}+2^{2}-2^{5}+2^{8}$ $-2^{10}-2^{13}-2^{16}-2^{22}$ $+2^{27}-2^{29}+2^{31}$	$-$	$-2^{1}+2^{3}+2^{5}+2^{8}$ $+2^{10}-2^{13}+2^{17}$ $+2^{25}+2^{27}-2^{29}$	22
8	$+2^{2}-2^{4}+2^{8}+2^{15}+2^{17}$ $-2^{19}-2^{23}+2^{25}+2^{28}$	$2^{0}-2^{3}-2^{13}-2^{15}-2^{17}$ $+2^{21}+2^{23}-2^{26}+2^{30}$	$-$	$-2^{1}-2^{8}-2^{10}+2^{12}$ $+2^{16}-2^{18}-2^{21}-2^{23}$ $-2^{25}+2^{29}+2^{31}$	7
9	$-2^{0}+2^{2}-2^{6}-2^{30}$	$-2^{0}+2^{8}-2^{19}$ $-2^{23}-2^{27}+2^{29}$	$-$	$2^{0}+2^{19}-2^{22}+2^{26}$	12
10	$+2^{12}+2^{30}$	$-2^{0}+2^{2}+2^{28}$	$-$	$-2^{1}+2^{7}-2^{27}-2^{29}$	17
11	$-2^{14}-2^{18}+2^{24}+2^{30}$	2^{2}	-2^{15}	$-2^{3}+2^{8}+2^{17}-2^{19}$ $-2^{23}+2^{25}+2^{28}$	22
12	$+2^{7}-2^{9}+2^{13}-2^{24}-2^{31}$	2^{30}	$-$	$-2^{0}+2^{2}-2^{6}$	7
13	$-2^{24}-2^{31}$	-2^{30}	$-$	2^{12}	12
14	$+2^{31}$	$-2^{24}+2^{31}$	2^{31}	$-2^{14}-2^{18}+2^{30}$	17
15	$-2^{3}+2^{15}$	$-2^{24}+2^{31}$	$-$	$2^{7}-2^{9}+2^{13}-2^{25}$	22
16	$+2^{29}-2^{31}$	2^{31}	$-$	-2^{24}	5
17	$+2^{31}$	2^{31}	$-$	$-$	9
18	-2^{31}	$-$	-2^{15}	-2^{3}	14
19	$-2^{17}-2^{31}$	2^{31}	$-$	2^{29}	20
20	-2^{31}	2^{31}	$-$	$-$	5
21	-2^{31}	2^{31}	$-$	$-$	9
22	-2^{31}	2^{31}	$-$	-2^{17}	14
23	$-$	$-$	2^{31}	$-$	20
24	$-$	2^{31}	$-$	$-$	5
25	$-$	$-$	2^{31}	$-$	9

Table B-10: New second block conditions nr. 2

t	Conditions on Q_t: $b_{31} \dots b_0$	#
-2	`......0. ........ ........ ........`	(1)
-1	`^.....0. ........ ........ .......0`	(3)
0	`^.....0. ........ ........ .1.....0`	(4)
	Total # IV conditions for 1st block	(8)
1	`^.....0. ......0. ....1..1 .1.....0`	7
2	`^....00. .1..1100 ..111..0 .0...0.0`	15
3	`^01..10. 00..0001 000000.1 ^01.11^0`	25
4	`011.001^ 10..0111 11011010 11^^1011`	29
5	`000.0010 10^^1001 10010101 01011100`	31
6	`10010001 10011011 00000000 01100110`	32
7	`01.00001 0.001.01 0.011110 01100101`	28
8	`1010.00. 11011.00 00011110 01111001`	29
9	`01.10.00 1...1.0. 01.10..1 11.0.0.1`	19
10	`00.0..10 1...1^1. 11.01..1 .0.1...1`	17
11	`00!....0 .....1.. .1^00.^. ^1...0.1`	14
12	`10.....1 ....!0.. .001..1. 0.......`	10
13	`10.....1 .....1.. 110...0. 0...1...`	10
14	`0.0..... ........ 1.1...1. 1...1...`	7
15	`111....0 .......! 0....... ....1...`	7
16	`1.0..... ......!. ........ ........`	3
17	`0....... ......0. ^....... ....^...`	4
18	`1.^..... ......1. ........ ........`	3
19	`1....... ......1. ........ ........`	2
20	`1....... .....!.. ........ ........`	2
21	`1....... ......^. ........ ........`	2
22	`1....... ........ ........ ........`	1
23	`0....... ........ ........ ........`	1
24	`1....... ........ ........ ........`	1
	Subtotal # conditions	299

B.4.3 New Second Block Differential Path nr. 3

Table B-11: New second block differential path nr. 3

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
-3	$+2^{31}$	$\times$	$\times$	$\times$	$\times$
-2	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
-1	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
0	$+2^{25}+2^{31}$	$2^{25}+2^{31}$	$-$	2^{25}	7
1	$+2^{0}+2^{25}+2^{31}$	$2^{25}+2^{31}$	$-$	2^{26}	12
2	$+2^{0}+2^{6}+2^{25}+2^{31}$	$2^{25}+2^{31}$	$-$	2^{26}	17
3	$+2^{0}+2^{6}+2^{11}+2^{25}+2^{31}$	$2^{0}+2^{6}+2^{25}+2^{31}$	$-$	$2^{0}+2^{6}+2^{26}$	22
4	$-2^{0}+2^{1}, -2^{6}\dots-2^{10}$ $+2^{12}-2^{16}-2^{17}-2^{18}$ $+2^{19}-2^{22}+2^{23}+2^{25}$ $-2^{28}+2^{29}-2^{31}$	$2^{0}-2^{6}+2^{9}-2^{11}$ $+2^{13}+2^{17}+2^{19}+2^{22}$ $+2^{25}-2^{29}+2^{31}$	2^{31}	$2^{1}-2^{6}+2^{9}-2^{11}$ $+2^{13}+2^{17}+2^{19}+2^{22}$ $+2^{26}-2^{29}+2^{31}$	7
5	$-2^{0}+2^{2}+2^{4}-2^{5}+2^{8}+2^{11}$ $+2^{13}+2^{14}+2^{15}-2^{16}+2^{17}$ $+2^{18}-2^{19}-2^{20}+2^{21}+2^{22}$ $-2^{24}+2^{27}-2^{28}+2^{30}+2^{31}$	$2^{1}-2^{5}+2^{11}-2^{14}$ $+2^{18}-2^{21}+2^{24}+2^{30}$	$-$	$-2^{0}+2^{2}+2^{5}+2^{11}$ $-2^{14}+2^{18}-2^{21}$ $-2^{24}+2^{26}-2^{30}$	12
6	$-2^{0}\dots-2^{3}+2^{4}, -2^{5}-2^{6}$ $+2^{7}+2^{8}+2^{10}, -2^{12}\dots-2^{18}$ $+2^{19}+2^{20}-2^{22}+2^{26}-2^{28}$	$-2^{0}-2^{3}-2^{5}+2^{9}$ $+2^{11}-2^{13}-2^{20}-2^{23}$ $+2^{26}-2^{28}+2^{31}$	$-$	$-2^{3}+2^{5}+2^{9}-2^{12}$ $-2^{20}-2^{23}-2^{25}-2^{27}$	17
7	$+2^{0}-2^{27}-2^{29}$	$2^{0}+2^{3}+2^{5}+2^{7}+2^{10}$ $+2^{12}-2^{14}+2^{16}-2^{19}$ $-2^{22}-2^{27}-2^{29}+2^{31}$	$-$	$2^{1}+2^{3}-2^{5}+2^{8}$ $-2^{10}-2^{13}+2^{17}-2^{19}$ $+2^{25}+2^{27}-2^{29}$	22
8	$-2^{3}+2^{7}-2^{9}+2^{15}+2^{17}$ $-2^{19}+2^{23}+2^{25}+2^{28}$	$-2^{0}+2^{4}+2^{9}+2^{13}$ $+2^{15}+2^{17}+2^{19}$ $+2^{23}-2^{25}-2^{29}$	$-$	$2^{1}-2^{8}-2^{10}+2^{12}$ $+2^{15}-2^{19}-2^{21}-2^{25}$ $-2^{27}+2^{29}+2^{31}$	7
9	$-2^{0}+2^{2}-2^{6}-2^{22}-2^{24}$	$2^{7}-2^{9}+2^{28}$	$-$	$2^{0}+2^{5}-2^{7}+2^{10}$ $+2^{12}+2^{20}-2^{22}+2^{26}$	12
10	$+2^{12}+2^{17}-2^{19}$	$2^{7}-2^{12}$	$-$	$2^{0}+2^{7}-2^{12}-2^{27}-2^{29}$	17
11	$-2^{14}-2^{18}+2^{24}-2^{29}$	-2^{24}	-2^{15}	$-2^{3}+2^{7}-2^{9}+2^{17}-2^{19}$ $-2^{23}+2^{25}+2^{28}$	22
12	$+2^{7}-2^{9}+2^{13}-2^{24}-2^{31}$	$-$	$-$	$-2^{0}+2^{2}-2^{6}-2^{22}-2^{24}$	7
13	$-2^{24}-2^{29}$	$-$	$-$	$2^{12}+2^{17}-2^{19}$	12
14	-2^{31}	$-2^{24}-2^{29}$	2^{31}	$-2^{14}-2^{18}+2^{30}$	17
15	$-2^{3}+2^{15}$	$-2^{24}+2^{31}$	$-$	$2^{7}-2^{9}+2^{13}-2^{25}$	22
16	$+2^{29}-2^{31}$	2^{29}	$-$	-2^{24}	5
17	$+2^{31}$	2^{31}	$-$	$-$	9
18	-2^{31}	$-$	-2^{15}	-2^{3}	14
19	$-2^{17}-2^{31}$	2^{31}	$-$	2^{29}	20
20	-2^{31}	2^{31}	$-$	$-$	5
21	-2^{31}	2^{31}	$-$	$-$	9
22	-2^{31}	2^{31}	$-$	-2^{17}	14
23	$-$	$-$	2^{31}	$-$	20
24	$-$	2^{31}	$-$	$-$	5
25	$-$	$-$	2^{31}	$-$	9

Table B-12: New second block conditions nr. 3

t	Conditions on Q_t: $b_{31} \dots b_0$	#
-2	`......0. ........ ........ ........`	(1)
-1	`^.....0. ........ ........ .......1`	(3)
0	`^.....0. ........ ........ .0.....1`	(4)
	Total # IV conditions for 1st block	(8)
1	`^.....0. ........ ....00.. .0!....0`	7
2	`^.11..0. .1..0110 ...00100 00.....0`	17
3	`^001..00 ^0111100 00110010 1011.0^0`	29
4	`!101^001 01010111 11101111 11010001`	32
5	`.00101.1 00011001 00000000 10101011`	31
6	`.0110011 01000111 11110000 01101111`	31
7	`01111001 1110101. 1001111. 01..0110`	28
8	`1.100101 01.01000 0001.011 01101101`	29
9	`..111.01 01..0.0. 0..0..1. 111100.1`	19
10	`1011..10 10..1^0. 1^.0..1. 00..10.0`	20
11	`111....0 .1..010! .1^0..1. 01...1.1`	17
12	`100....1 ....101. .001..1. 0.......`	12
13	`011....1 .....1.. 110...0. 0...1...`	11
14	`111..... ........ 1.1...1. 1...1...`	8
15	`101....0 .......! 0....... ....1...`	7
16	`100..... ......!. ........ ........`	4
17	`0....... ......0. ^....... ....^...`	4
18	`1.^..... ......1. ........ ........`	3
19	`1....... ......1. ........ ........`	2
20	`1....... .....!.. ........ ........`	2
21	`1....... ......^. ........ ........`	2
22	`1....... ........ ........ ........`	1
23	`0....... ........ ........ ........`	1
24	`1....... ........ ........ ........`	1
	Subtotal # conditions	318

B.4.4 New Second Block Differential Path nr. 4

Table B-13: New second block differential path nr. 4

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
-3	$+2^{31}$	$\times$	$\times$	$\times$	$\times$
-2	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
-1	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
0	$+2^{25}+2^{31}$	$2^{25}+2^{31}$	–	2^{25}	7
1	$+2^{0}+2^{25}+2^{31}$	$2^{25}+2^{31}$	–	2^{26}	12
2	$+2^{0}+2^{6}+2^{25}+2^{31}$	$2^{0}+2^{25}+2^{31}$	–	$2^{0}+2^{26}$	17
3	$+2^{0}+2^{6}+2^{11}$ $+2^{17}+2^{25}+2^{31}$	$2^{0}+2^{6}-2^{11}$ $+2^{25}+2^{31}$	–	$2^{0}+2^{6}-2^{11}+2^{26}$	22
4	$-2^{0}-2^{6}+2^{7}, -2^{11}\ldots-2^{14}$ $+2^{15}-2^{16}-2^{18}-2^{19}$ $+2^{20}+2^{22}-2^{25}+2^{26}$ $-2^{28}+2^{29}-2^{31}$	$2^{0}-2^{6}-2^{12}-2^{16}$ $+2^{18}-2^{20}+2^{22}$ $-2^{25}-2^{27}+2^{31}$	2^{31}	$2^{1}-2^{6}-2^{12}-2^{16}$ $+2^{18}-2^{20}+2^{22}$ $-2^{27}+2^{31}$	7
5	$+2^{0}+2^{1}+2^{3}+2^{4}+2^{5}-2^{6}$ $-2^{8}+2^{9}-2^{11}-2^{12}+2^{16}-2^{17}$ $+2^{18}\ldots+2^{21}, +2^{23}+2^{24}$ $+2^{25}-2^{27}-2^{28}+2^{30}-2^{31}$	$2^{1}-2^{4}+2^{8}-2^{10}$ $+2^{15}+2^{18}-2^{21}+2^{23}$ $-2^{26}-2^{28}+2^{30}$	–	$-2^{0}+2^{2}-2^{4}+2^{6}+2^{8}$ $-2^{10}+2^{15}+2^{18}-2^{21}$ $+2^{23}-2^{25}-2^{28}-2^{30}$	12
6	$-2^{0}+2^{1}-2^{5}-2^{10}-2^{11}$ $-2^{13}-2^{14}+2^{15}-2^{17}-2^{20}$ $+2^{21}+2^{23}+2^{24}-2^{25}$ $+2^{26}+2^{28}-2^{29}$	$-2^{3}+2^{9}+2^{11}-2^{13}$ $-2^{15}-2^{17}+2^{20}$ $+2^{26}-2^{28}+2^{31}$	–	$2^{0}-2^{3}+2^{6}+2^{9}-2^{12}$ $-2^{15}+2^{20}-2^{25}-2^{27}$	17
7	$-2^{27}-2^{29}$	$-2^{0}-2^{2}-2^{4}-2^{6}+2^{8}$ $-2^{11}-2^{13}-2^{16}-2^{19}$ $-2^{22}-2^{27}-2^{29}+2^{31}$	–	$2^{1}+2^{3}-2^{5}+2^{8}$ $-2^{13}+2^{17}-2^{19}$ $+2^{25}+2^{27}-2^{29}$	22
8	$-2^{3}+2^{7}-2^{9}+2^{15}+2^{17}$ $-2^{19}+2^{23}+2^{25}+2^{28}$	$-2^{0}+2^{3}+2^{9}+2^{13}$ $-2^{15}-2^{17}+2^{21}$ $+2^{25}+2^{27}-2^{29}$	–	$2^{1}-2^{8}-2^{10}+2^{12}$ $+2^{15}-2^{19}-2^{21}-2^{25}$ $-2^{27}+2^{29}+2^{31}$	7
9	$-2^{0}+2^{2}-2^{6}-2^{22}-2^{24}$	$2^{17}+2^{22}+2^{28}$	–	$2^{0}-2^{5}+2^{10}+2^{12}$ $+2^{20}-2^{22}+2^{26}$	12
10	$+2^{12}-2^{17}$	$-2^{0}+2^{7}-2^{12}$	–	$-2^{0}+2^{7}-2^{12}$ $-2^{27}-2^{29}$	17
11	$-2^{14}-2^{18}+2^{24}-2^{29}$	-2^{24}	-2^{15}	$-2^{3}+2^{7}-2^{9}+2^{17}$ $-2^{19}-2^{23}+2^{25}+2^{28}$	22
12	$+2^{7}-2^{9}+2^{13}-2^{24}+2^{31}$	–	–	$-2^{0}+2^{2}-2^{6}$ $-2^{22}-2^{24}$	7
13	$-2^{24}-2^{29}$	-2^{18}	–	$2^{12}+2^{17}-2^{19}$	12
14	$+2^{31}$	$-2^{24}-2^{29}$	2^{31}	$-2^{14}-2^{18}+2^{30}$	17
15	$-2^{3}+2^{15}$	$-2^{24}+2^{31}$	–	$2^{7}-2^{9}+2^{13}-2^{25}$	22
16	$+2^{29}+2^{31}$	2^{29}	–	-2^{24}	5
17	-2^{31}	2^{31}	–	–	9
18	$+2^{31}$	–	-2^{15}	-2^{3}	14
19	$-2^{17}+2^{31}$	2^{31}	–	2^{29}	20
20	$+2^{31}$	2^{31}	–	–	5
21	$+2^{31}$	2^{31}	–	–	9
22	$+2^{31}$	2^{31}	–	-2^{17}	14
23	–	–	2^{31}	–	20
24	–	2^{31}	–	–	5
25	–	–	2^{31}	–	9

Table B-14: New second block conditions nr. 4

t	Conditions on Q_t: $b_{31} \dots b_0$	#
-2	`......0. ........ ........ ........`	(1)
-1	`^.....0. ........ ........ .......0`	(3)
0	`^.....0. ........ ........ .0.....0`	(4)
	Total # IV conditions for 1st block	(8)
1	`^.....0. ......1. ....1... .0.....0`	6
2	`^.10.00. .0.11111 ...00... 10.....0`	16
3	`^0111101 11001100 ^^^10.10 0011..00`	29
4	`!0010011 10101111 01111001 0110^.11`	31
5	`10111100 01000010 10011001 01000.00`	31
6	`00100010 01010111 011.1110 1.111.01`	29
7	`10111101 00.00100 0011.000 10110..0`	28
8	`0..01001 01011.0. 0001111. 001.1^10`	26
9	`1.111.01 11..0.1. 0..0..0. 010.00.1`	19
10	`1111..10 10..1^1. 1^.0..1. 00..10.1`	20
11	`101....0 .1...10! .1^0..1. 01...1.1`	16
12	`010....1 ..!..01. .001..1. 0.......`	12
13	`0011...1 .....0.. 110...0. 0...1...`	12
14	`0010.... ........ 1.1...1. 1...1...`	9
15	`1110...0 .......! 0....... ....1...`	8
16	`0101.... ......!. ........ ........`	5
17	`1....... ......0. ^....... ....^...`	4
18	`0.^..... ......1. ........ ........`	3
19	`0....... ......1. ........ ........`	2
20	`0....... .....!.. ........ ........`	2
21	`0....... ......^. ........ ........`	2
22	`0....... ........ ........ ........`	1
23	`0....... ........ ........ ........`	1
24	`1....... ........ ........ ........`	1
	Subtotal # conditions	313

C Boolean Function Bitconditions

C.1 Bitconditions applied to boolean function F

Table C-1: Bitconditions applied to boolean function F

$$F(X,Y,Z) = (X \wedge Y) \oplus (\bar{X} \wedge Z)$$

DB	Forward bitconditions			Backward bitconditions		
abc	$g=0$	$g=+1$	$g=-1$	$g=0$	$g=+1$	$g=-1$
`...` (8)	`...` (8)			`...` (8)		
`..+` (4)	`1.+` (2)	`0.+` (2)		`1.+` (2)	`0.+` (2)	
`..-` (4)	`1.-` (2)		`0.-` (2)	`1.-` (2)		`0.-` (2)
`.+.` (4)	`0+.` (2)	`1+.` (2)		`0+.` (2)	`1+.` (2)	
`.++` (2)		`.++` (2)			`.++` (2)	
`.+-` (2)		`1+-` (1)	`0+-` (1)		`1+-` (1)	`0+-` (1)
`.-.` (4)	`0-.` (2)		`1-.` (2)	`0-.` (2)		`1-.` (2)
`.-+` (2)		`0-+` (1)	`1-+` (1)		`0-+` (1)	`1-+` (1)
`.--` (2)			`.--` (2)			`.--` (2)
`+..` (4)	`+.V` (2)	`+10` (1)	`+01` (1)	`+^.` (2)	`+10` (1)	`+01` (1)
`+.+` (2)	`+0+` (1)	`+1+` (1)		`+0+` (1)	`+1+` (1)	
`+.-` (2)	`+1-` (1)		`+0-` (1)	`+1-` (1)		`+0-` (1)
`++.` (2)	`++1` (1)	`++0` (1)		`++1` (1)	`++0` (1)	
`+++` (1)		`+++` (1)			`+++` (1)	
`++-` (1)	`++-` (1)			`++-` (1)		
`+-.` (2)	`+-0` (1)		`+-1` (1)	`+-0` (1)		`+-1` (1)
`+-+` (1)	`+-+` (1)			`+-+` (1)		
`+--` (1)			`+--` (1)			`+--` (1)
`-..` (4)	`-.V` (2)	`-01` (1)	`-10` (1)	`-^.` (2)	`-01` (1)	`-10` (1)
`-.+` (2)	`-1+` (1)	`-0+` (1)		`-1+` (1)	`-0+` (1)	
`-.-` (2)	`-0-` (1)		`-1-` (1)	`-0-` (1)		`-1-` (1)
`-+.` (2)	`-+0` (1)	`-+1` (1)		`-+0` (1)	`-+1` (1)	
`-++` (1)		`-++` (1)			`-++` (1)	
`-+-` (1)	`-+-` (1)			`-+-` (1)		
`--.` (2)	`--1` (1)		`--0` (1)	`--1` (1)		`--0` (1)
`--+` (1)	`--+` (1)			`--+` (1)		
`---` (1)			`---` (1)			`---` (1)

Here abc denotes three bitconditions $(\mathfrak{q}_t[i], \mathfrak{q}_{t-1}[i], \mathfrak{q}_{t-2}[i])$ for $0 \le t \le 15$ and $0 \le i \le 31$. The next three columns hold the forward bitconditions $FC(t, abc, 0)$, $FC(t, abc, +1)$ and $FC(t, abc, -1)$, respectively. The last three columns hold the backward bitconditions $BC(t, abc, 0)$, $BC(t, abc, +1)$ and $BC(t, abc, -1)$, respectively.
Next to each triple of bitconditions def is denoted $|U_{def}|$, the amount of freedom left.
An entry is left empty if $g \notin V_{abc}$. See subsubsection 6.3.2 for more details.

C.2 Bitconditions applied to boolean function G

Table C-2: Bitconditions applied to MD5 boolean function G

$$G(X,Y,Z) = (Z \wedge X) \oplus (\bar{Z} \wedge Y)$$

DB	Forward bitconditions			Backward bitconditions		
abc	$g = 0$	$g = +1$	$g = -1$	$g = 0$	$g = +1$	$g = -1$
`...` (8)	`...` (8)			`...` (8)		
`..+` (4)	`.V+` (2)	`10+` (1)	`01+` (1)	`^.+` (2)	`10+` (1)	`01+` (1)
`..-` (4)	`.V-` (2)	`01-` (1)	`10-` (1)	`^.-` (2)	`01-` (1)	`10-` (1)
`.+.` (4)	`.+1` (2)	`.+0` (2)		`.+1` (2)	`.+0` (2)	
`.++` (2)	`0++` (1)	`1++` (1)		`0++` (1)	`1++` (1)	
`.+-` (2)	`1+-` (1)	`0+-` (1)		`1+-` (1)	`0+-` (1)	
`.-.` (4)	`.-1` (2)		`.-0` (2)	`.-1` (2)		`.-0` (2)
`.-+` (2)	`1-+` (1)		`0-+` (1)	`1-+` (1)		`0-+` (1)
`.--` (2)	`0--` (1)		`1--` (1)	`0--` (1)		`1--` (1)
`+..` (4)	`+.0` (2)	`+.1` (2)		`+.0` (2)	`+.1` (2)	
`+.+` (2)	`+1+` (1)	`+0+` (1)		`+1+` (1)	`+0+` (1)	
`+.-` (2)	`+0-` (1)	`+1-` (1)		`+0-` (1)	`+1-` (1)	
`++.` (2)		`++.` (2)			`++.` (2)	
`+++` (1)		`+++` (1)			`+++` (1)	
`++-` (1)		`++-` (1)			`++-` (1)	
`+-.` (2)		`+-1` (1)	`+-0` (1)		`+-1` (1)	`+-0` (1)
`+-+` (1)	`+-+` (1)			`+-+` (1)		
`+--` (1)	`+--` (1)			`+--` (1)		
`-..` (4)	`-.0` (2)		`-.1` (2)	`-.0` (2)		`-.1` (2)
`-.+` (2)	`-0+` (1)		`-1+` (1)	`-0+` (1)		`-1+` (1)
`-.-` (2)	`-1-` (1)		`-0-` (1)	`-1-` (1)		`-0-` (1)
`-+.` (2)		`-+0` (1)	`-+1` (1)		`-+0` (1)	`-+1` (1)
`-++` (1)	`-++` (1)			`-++` (1)		
`-+-` (1)	`-+-` (1)			`-+-` (1)		
`--.` (2)			`--.` (2)			`--.` (2)
`--+` (1)			`--+` (1)			`--+` (1)
`---` (1)			`---` (1)			`---` (1)

Here abc denotes three bitconditions $(\mathfrak{q}_t[i], \mathfrak{q}_{t-1}[i], \mathfrak{q}_{t-2}[i])$ for $16 \leq t \leq 31$ and $0 \leq i \leq 31$. The next three columns hold the forward bitconditions $FC(t, abc, 0)$, $FC(t, abc, +1)$ and $FC(t, abc, -1)$, respectively. The last three columns hold the backward bitconditions $BC(t, abc, 0)$, $BC(t, abc, +1)$ and $BC(t, abc, -1)$, respectively.
Next to each triple of bitconditions def is denoted $|U_{def}|$, the amount of freedom left.
An entry is left empty if $g \notin V_{abc}$. See subsubsection 6.3.2 for more details.

C.3 Bitconditions applied to boolean function H

Table C-3: Bitconditions applied to MD5 boolean function H

$$H(X,Y,Z) = X \oplus Y \oplus Z$$

DB	Forward bitconditions			Backward bitconditions		
abc	$g=0$	$g=+1$	$g=-1$	$g=0$	$g=+1$	$g=-1$
... (8)	... (8)			... (8)		
..+ (4)		.V+ (2)	.Y+ (2)		^.+ (2)	!.+ (2)
..- (4)		.Y- (2)	.V- (2)		!.- (2)	^.- (2)
.+. (4)		.+W (2)	.+H (2)		m+. (2)	#+. (2)
.++ (2)	.++ (2)			.++ (2)		
.+- (2)	.+- (2)			.+- (2)		
.-. (4)		.-H (2)	.-W (2)		#-. (2)	m-. (2)
.-+ (2)	.-+ (2)			.-+ (2)		
.-- (2)	.-- (2)			.-- (2)		
+.. (4)		+.V (2)	+.Y (2)		+^. (2)	+!. (2)
+.+ (2)	+.+ (2)			+.+ (2)		
+.- (2)	+.- (2)			+.- (2)		
++. (2)	++. (2)			++. (2)		
+++ (1)		+++ (1)			+++ (1)	
++- (1)			++- (1)			++- (1)
+-. (2)	+-. (2)			+-. (2)		
+-+ (1)			+-+ (1)			+-+ (1)
+-- (1)		+-- (1)			+-- (1)	
-.. (4)		-.Y (2)	-.V (2)		-!. (2)	-^. (2)
-.+ (2)	-.+ (2)			-.+ (2)		
-.- (2)	-.- (2)			-.- (2)		
-+. (2)	-+. (2)			-+. (2)		
-++ (1)			-++ (1)			-++ (1)
-+- (1)		-+- (1)			-+- (1)	
--. (2)	--. (2)			--. (2)		
--+ (1)		--+ (1)			--+ (1)	
--- (1)			--- (1)			--- (1)

Here *abc* denotes three bitconditions $(\mathtt{q}_t[i], \mathtt{q}_{t-1}[i], \mathtt{q}_{t-2}[i])$ for $32 \leq t \leq 47$ and $0 \leq i \leq 31$. The next three columns hold the forward bitconditions $FC(t, abc, 0)$, $FC(t, abc, +1)$ and $FC(t, abc, -1)$, respectively. The last three columns hold the backward bitconditions $BC(t, abc, 0)$, $BC(t, abc, +1)$ and $BC(t, abc, -1)$, respectively.
Next to each triple of bitconditions def is denoted $|U_{def}|$, the amount of freedom left.
An entry is left empty if $g \notin V_{abc}$. See subsubsection 6.3.2 for more details.

C.4 Bitconditions applied to boolean function I

Table C-4: Bitconditions applied to MD5 boolean function I

$$I(X,Y,Z) = Y \oplus (X \vee \bar{Z})$$

DB	Forward bitconditions			Backward bitconditions		
abc	$g=0$	$g=+1$	$g=-1$	$g=0$	$g=+1$	$g=-1$
`...` (8)	`...` (8)			`...` (8)		
`..+` (4)	`1.+` (2)	`01+` (1)	`00+` (1)	`1.+` (2)	`01+` (1)	`00+` (1)
`..-` (4)	`1.-` (2)	`00-` (1)	`01-` (1)	`1.-` (2)	`00-` (1)	`01-` (1)
`.+.` (4)		`0+1` (1)	`.+Q` (3)		`0+1` (1)	`?+.` (3)
`.++` (2)	`0++` (1)		`1++` (1)	`0++` (1)		`1++` (1)
`.+-` (2)	`0+-` (1)		`1+-` (1)	`0+-` (1)		`1+-` (1)
`.-.` (4)		`.-Q` (3)	`0-1` (1)		`?-.` (3)	`0-1` (1)
`.-+` (2)	`0-+` (1)	`1-+` (1)		`0-+` (1)	`1-+` (1)	
`.--` (2)	`0--` (1)	`1--` (1)		`0--` (1)	`1--` (1)	
`+..` (4)	`+.0` (2)	`+01` (1)	`+11` (1)	`+.0` (2)	`+01` (1)	`+11` (1)
`+.+` (2)	`+.+` (2)			`+.+` (2)		
`+.-` (2)		`+0-` (1)	`+1-` (1)		`+0-` (1)	`+1-` (1)
`++.` (2)	`++1` (1)		`++0` (1)	`++1` (1)		`++0` (1)
`+++` (1)			`+++` (1)			`+++` (1)
`++-` (1)	`++-` (1)			`++-` (1)		
`+-.` (2)	`+-1` (1)	`+-0` (1)		`+-1` (1)	`+-0` (1)	
`+-+` (1)		`+-+` (1)			`+-+` (1)	
`+--` (1)	`+--` (1)			`+--` (1)		
`-..` (4)	`-.0` (2)	`-11` (1)	`-01` (1)	`-.0` (2)	`-11` (1)	`-01` (1)
`-.+` (2)		`-1+` (1)	`-0+` (1)		`-1+` (1)	`-0+` (1)
`-.-` (2)	`-.-` (2)			`-.-` (2)		
`-+.` (2)	`-+1` (1)		`-+0` (1)	`-+1` (1)		`-+0` (1)
`-++` (1)	`-++` (1)			`-++` (1)		
`-+-` (1)			`-+-` (1)			`-+-` (1)
`--.` (2)	`--1` (1)	`--0` (1)		`--1` (1)	`--0` (1)	
`--+` (1)	`--+` (1)			`--+` (1)		
`---` (1)		`---` (1)			`---` (1)	

Here *abc* denotes three bitconditions $(\mathfrak{q}_t[i], \mathfrak{q}_{t-1}[i], \mathfrak{q}_{t-2}[i])$ for $48 \leq t \leq 63$ and $0 \leq i \leq 31$. The next three columns hold the forward bitconditions $FC(t, abc, 0)$, $FC(t, abc, +1)$ and $FC(t, abc, -1)$, respectively. The last three columns hold the backward bitconditions $BC(t, abc, 0)$, $BC(t, abc, +1)$ and $BC(t, abc, -1)$, respectively.
Next to each triple of bitconditions def is denoted $|U_{def}|$, the amount of freedom left. An entry is left empty if $g \notin V_{abc}$. See subsubsection 6.3.2 for more details.

D Chosen-Prefix Collision Example - Colliding Certificates

D.1 Chosen Prefixes

Table D-1: Chosen Prefix 1: Partial X.509 Certificate with identity *Arjen K. Lenstra*

P_{AL} =

```
30 82 05 11 A0 03 02 01 02 02 04 01 0C 00 01 30
0D 06 09 2A 86 48 86 F7 0D 01 01 04 05 00 30 3D
31 1A 30 18 06 03 55 04 03 13 11 48 61 73 68 20
43 6F 6C 6C 69 73 69 6F 6E 20 43 41 31 12 30 10

06 03 55 04 07 13 09 45 69 6E 64 68 6F 76 65 6E
31 0B 30 09 06 03 55 04 06 13 02 4E 4C 30 1E 17
0D 30 36 30 31 30 31 30 30 30 30 30 31 5A 17 0D
30 37 31 32 33 31 32 33 35 39 35 39 5A 30 54 31

19 30 17 06 03 55 04 03 13 10 41 72 6A 65 6E 20
4B 2E 20 4C 65 6E 73 74 72 61 31 16 30 14 06 03
55 04 0A 13 0D 43 6F 6C 6C 69 73 69 6F 6E 61 69
72 73 31 12 30 10 06 03 55 04 07 13 09 45 69 6E

64 68 6F 76 65 6E 31 0B 30 09 06 03 55 04 06 13
02 4E 4C 30 82 04 22 30 0D 06 09 2A 86 48 86 F7
0D 01 01 01 05 00 03 82 04 0F 00 30 82 04 0A 02
82 04 01 00
```

$$IHV_{\mathrm{AL}} = IHV_3 = \texttt{A2934A57268FC8FB99270DB2BD42867F}$$
$$= \{\texttt{574a93a2}_{16}, \texttt{fbc88f26}_{16}, \texttt{b20d2799}_{16}, \texttt{7f8642bd}_{16}\}$$

Table D-2: Chosen Prefix 2: Partial X.509 Certificate with identity *Marc Stevens*

P_{MS} =

```
30 82 05 11 A0 03 02 01 02 02 04 02 0C 00 01 30
0D 06 09 2A 86 48 86 F7 0D 01 01 04 05 00 30 3D
31 1A 30 18 06 03 55 04 03 13 11 48 61 73 68 20
43 6F 6C 6C 69 73 69 6F 6E 20 43 41 31 12 30 10

06 03 55 04 07 13 09 45 69 6E 64 68 6F 76 65 6E
31 0B 30 09 06 03 55 04 06 13 02 4E 4C 30 1E 17
0D 30 36 30 31 30 31 30 30 30 30 30 31 5A 17 0D
30 37 31 32 33 31 32 33 35 39 35 39 5A 30 54 31

15 30 13 06 03 55 04 03 13 0C 4D 61 72 63 20 53
74 65 76 65 6E 73 31 1A 30 18 06 03 55 04 0A 13
11 43 6F 6C 6C 69 73 69 6F 6E 20 46 61 63 74 6F
72 79 31 12 30 10 06 03 55 04 07 13 09 45 69 6E

64 68 6F 76 65 6E 31 0B 30 09 06 03 55 04 06 13
02 4E 4C 30 82 04 22 30 0D 06 09 2A 86 48 86 F7
0D 01 01 01 05 00 03 82 04 0F 00 30 82 04 0A 02
82 04 01 00
```

$$IHV_{\mathrm{MS}} = IHV_3 = \texttt{9756EBE66FC92AD60256345C8EC444A8}$$
$$= \{\texttt{e6eb5697}_{16}, \texttt{d62ac96f}_{16}, \texttt{5c345602}_{16}, \texttt{a844c48e}_{16}\}$$

Table D-3: Chosen Prefix 3: Partial X.509 Certificate with identity *Benne de Weger*

$$P_{\mathrm{BW}} = $$

```
30 82 05 11 A0 03 02 01 02 02 04 03 0C 00 01 30
0D 06 09 2A 86 48 86 F7 0D 01 01 04 05 00 30 3D
31 1A 30 18 06 03 55 04 03 13 11 48 61 73 68 20
43 6F 6C 6C 69 73 69 6F 6E 20 43 41 31 12 30 10

06 03 55 04 07 13 09 45 69 6E 64 68 6F 76 65 6E
31 0B 30 09 06 03 55 04 06 13 02 4E 4C 30 1E 17
0D 30 36 30 31 30 31 30 30 30 30 30 31 5A 17 0D
30 37 31 32 33 31 32 33 35 39 35 39 5A 30 54 31

17 30 15 06 03 55 04 03 13 0E 42 65 6E 6E 65 20
64 65 20 57 65 67 65 72 31 18 30 16 06 03 55 04
0A 13 0F 43 6F 6C 6C 69 73 69 6F 6E 6D 61 6B 65
72 73 31 12 30 10 06 03 55 04 07 13 09 45 69 6E

64 68 6F 76 65 6E 31 0B 30 09 06 03 55 04 06 13
02 4E 4C 30 82 04 22 30 0D 06 09 2A 86 48 86 F7
0D 01 01 01 05 00 03 82 04 0F 00 30 82 04 0A 02
82 04 01 00
```

$$\begin{aligned} IHV_{\mathrm{BW}} = IHV_3 &= \texttt{5B2D26DB2317BE0A93D936FD47C7B013} \\ &= \{\texttt{db262d5b}_{16}, \texttt{0abe1723}_{16}, \texttt{fd36d993}_{16}, \texttt{13b0c747}_{16}\} \end{aligned}$$

D.2 Birthday attack

Table D-4: Birthday Attack

$$\begin{aligned} IHV_{\mathrm{AL}} &= \texttt{A2934A57268FC8FB99270DB2BD42867F} \\ &= \{\texttt{574a93a2}_{16}, \texttt{fbc88f26}_{16}, \texttt{b20d2799}_{16}, \texttt{7f8642bd}_{16}\} \\ IHV_{\mathrm{MS}} &= \texttt{9756EBE66FC92AD60256345C8EC444A8} \\ &= \{\texttt{e6eb5697}_{16}, \texttt{d62ac96f}_{16}, \texttt{5c345602}_{16}, \texttt{a844c48e}_{16}\} \\ IHV_{\mathrm{BW}} &= \texttt{5B2D26DB2317BE0A93D936FD47C7B013} \\ &= \{\texttt{db262d5b}_{16}, \texttt{0abe1723}_{16}, \texttt{fd36d993}_{16}, \texttt{13b0c747}_{16}\} \end{aligned}$$

$$R = $$

```
64 68 6F 76 65 6E 31 0B 30 09 06 03 55 04 06 13
02 4E 4C 30 82 04 22 30 0D 06 09 2A 86 48 86 F7
0D 01 01 01 05 00 03 82 04 0F 00 30 82 04 0A 02
82 04 01 00 xx xx xx xx yy yy yy yy zz zz zz zz
```

$$\begin{aligned} &= \{\, \texttt{766f6864}_{16},\ \texttt{0b316e65}_{16},\ \texttt{03060930}_{16},\ \texttt{13060455}_{16}, \\ &\quad \texttt{304c4e02}_{16},\ \texttt{30220482}_{16},\ \texttt{2a09060d}_{16},\ \texttt{f7864886}_{16}, \\ &\quad \texttt{0101010d}_{16},\ \texttt{82030005}_{16},\ \texttt{30000f04}_{16},\ \texttt{020a0482}_{16}, \\ &\quad \texttt{00010482}_{16},\quad \texttt{x},\quad \texttt{y},\quad \texttt{z} \quad \} \end{aligned}$$

$$\begin{aligned} \phi(x, y, z) &= \begin{cases} IHV_{\mathrm{AL}}, & \text{if } x = 0 \mod 3; \\ IHV_{\mathrm{MS}}, & \text{if } x = 1 \mod 3; \\ IHV_{\mathrm{BW}}, & \text{if } x = 2 \mod 3. \end{cases} \\ \psi(x, y, z) &= R \| x \| y \| z \\ \rho(IHV) = \rho(a, b, c, d) &= (a, d - b, d - c) \\ \\ \Phi(x, y, z) &= \rho(\mathrm{MD5Compress}(\phi(x, y, z), \psi(x, y, z))) \\ S &= \{(x, y, z) \mid (x \equiv 0 \mod 2^{15}) \wedge (RL(y, 15) \equiv 0 \mod 2^{15})\} \end{aligned}$$

Table D-5: Birthday attack - Results

$$
\begin{aligned}
IHV_3 = IHV_{\mathrm{MS}} &= \texttt{9756EBE66FC92AD60256345C8EC444A8} \\
&= \{\texttt{e6eb5697}_{16}, \texttt{d62ac96f}_{16}, \texttt{5c345602}_{16}, \texttt{a844c48e}_{16}\} \\
(X, Y, Z) &= (\texttt{cbb4091a}_{16},\ \texttt{7a26c740}_{16},\ \texttt{9b7f01af}_{16}) \\
IHV'_3 = IHV_{\mathrm{AL}} &= \texttt{A2934A57268FC8FB99270DB2BD42867F} \\
&= \{\texttt{574a93a2}_{16}, \texttt{fbc88f26}_{16}, \texttt{b20d2799}_{16}, \texttt{7f8642bd}_{16}\} \\
(X', Y', Z') &= (\texttt{d6e773ee}_{16},\ \texttt{ba4fb3b3}_{16},\ \texttt{023d39a1}_{16})
\end{aligned}
$$

M_4 = `64 68 6F 76 65 6E 31 0B 30 09 06 03 55 04 06 13`
`02 4E 4C 30 82 04 22 30 0D 06 09 2A 86 48 86 F7`
`0D 01 01 01 05 00 03 82 04 0F 00 30 82 04 0A 02`
`82 04 01 00 1A 09 B4 CB 40 C7 26 7A AF 01 7F 9B`

$$
\begin{aligned}
= \{ & \texttt{766f6864}_{16},\ \texttt{0b316e65}_{16},\ \texttt{03060930}_{16},\ \texttt{13060455}_{16}, \\
& \texttt{304c4e02}_{16},\ \texttt{30220482}_{16},\ \texttt{2a09060d}_{16},\ \texttt{f7864886}_{16}, \\
& \texttt{0101010d}_{16},\ \texttt{82030005}_{16},\ \texttt{30000f04}_{16},\ \texttt{020a0482}_{16}, \\
& \texttt{00010482}_{16},\ \texttt{cbb4091a}_{16},\ \texttt{7a26c740}_{16},\ \texttt{9b7f01af}_{16} \}
\end{aligned}
$$

M'_4 = `64 68 6F 76 65 6E 31 0B 30 09 06 03 55 04 06 13`
`02 4E 4C 30 82 04 22 30 0D 06 09 2A 86 48 86 F7`
`0D 01 01 01 05 00 03 82 04 0F 00 30 82 04 0A 02`
`82 04 01 00 EE 73 E7 D6 B3 B3 4F BA A1 39 3D 02`

$$
\begin{aligned}
= \{ & \texttt{766f6864}_{16},\ \texttt{0b316e65}_{16},\ \texttt{03060930}_{16},\ \texttt{13060455}_{16}, \\
& \texttt{304c4e02}_{16},\ \texttt{30220482}_{16},\ \texttt{2a09060d}_{16},\ \texttt{f7864886}_{16}, \\
& \texttt{0101010d}_{16},\ \texttt{82030005}_{16},\ \texttt{30000f04}_{16},\ \texttt{020a0482}_{16}, \\
& \texttt{00010482}_{16},\ \texttt{d6e773ee}_{16},\ \texttt{ba4fb3b3}_{16},\ \texttt{023d39a1}_{16} \}
\end{aligned}
$$

$$
\begin{aligned}
IHV_4 &= \texttt{2D857B4EA419FB613F17A61017126647} \\
&= \{\texttt{4e7b852d}_{16}, \texttt{61fb19a4}_{16}, \texttt{10a6173f}_{16}, \texttt{47661217}_{16}\} \\
IHV'_4 &= \texttt{2D857B4E0479B7259F7662D47771220B} \\
&= \{\texttt{4e7b852d}_{16}, \texttt{25b77904}_{16}, \texttt{d462769f}_{16}, \texttt{0b227177}_{16}\} \\
\delta IHV_4 &= \{0, \delta b_4, \delta b_4, \delta b_4\} \\
\delta b_4 &= -2^5 - 2^7 - 2^{13} + 2^{15} - 2^{18} - 2^{22} + 2^{26} - 2^{30}
\end{aligned}
$$

D.3 Differential Paths

D.3.1 Block 1 of 8

Table D-6: Differential Path - block 1

Using $\delta m_{11} = +2^{20}$

t	Bits Q_t: $b_{31} \dots b_0$	#
-3	01001110 01111011 10000101 00101101	32
-2	0-00+-11 0-100-10 0++100-+ 0++10111	32
-1	++010+00 -+100-10 0++1011- +0-11111	32
0	0-100+01 1-11-+11 0++11001 -0-00100	32
1	+0...+-. 1-+.++.- ....1 -.-.1+.0	16
2	1+...-+. 1.-.1..- .10^..!1 ..-^00.1	18
3	0+.0^-1. -1-011.- ^11+..^- 1.++01.+	25
4	11.1++1. -.01.... +0.0..-1 ..10+1.+	19
5	010-01.0 +00.0000 0-010001 000--00.	29
6	1-1+1+01 +1.01111 ++1+11++ 11.00-10	30
7	.0.+.0.- .1....-- 1+.-..+0 .-.11--.	17
8	.1...0.1 0-....00 100-..0+ 1+..0.-.	17
9	.0.10..1 .-....1+ .-1...++ 0+..101.	16
10	.10.0011 0.0000.0 01+00!1. +.00+.01	25
11	0010+11+ 10111111 1+0.11-0 111110+0	31
12	1-0.-1+1 0101..01 00+.00+1 001.100-	27
13	00+10-0- --+0^^+- 000^1+0- -1+.0-++	31
14	11000+-- -------- +.+----- +--1+---	31
15	+01-0100 0-1-10-0 -.1010-1 0+10100+	31
16	1001-11+ 01010001 +.000000 000+100-	31
17	1.10...0 .1.0..-. 1.1...+. .0.-..10	14
18	..+.^..- ...1.... +.-...+. ...1..+0	10
19	^^.....- .0.+..^. +....0-. ...-....	10
20	..^....0 01...0.. 0.^..1-. .0...0^.	12
21	..0....1 1-.^.1.. ^....-0. .1.^.1..	12
22	..1..... +....+.. ......^. .+...-..	6
23	..+.0... .^...... 0....^..	5
24	....1... ^....^.. 1...0... .^...^.0	8
25	..^.-... .... -...1...	4
26	....0.... ....-... ......0+	4
27	....^... ...1.... ^....... ......1+	5
28	....+.... ....^... ......+-	4
29	....0.... 0	2
30	....-.... ....^1	3
31	....-....	1
32	....	0
33	....!....	1
$34-60$	....	0
61	....	
62	.+......	
63	.+......	
64	.+......	

Table D-7: Block 1 found using path in Table D-6

$$
\begin{aligned}
M_5 &= \texttt{A4 74 25 81 8D C8 4F 86 73 6E 90 72 28 BB E8 77} \\
&\ \texttt{02 03 85 8D 8C F1 83 7A FF 5E 6C 22 13 03 6A F3} \\
&\ \texttt{D9 5C 77 E9 C2 23 7D 60 8C C4 A9 FB 97 30 7B BF} \\
&\ \texttt{98 28 61 2F 15 99 E2 61 5B CC DE DA 59 30 53 2F} \\
&= \{\ \texttt{812574a4}_{16},\ \texttt{864fc88d}_{16},\ \texttt{72906e73}_{16},\ \texttt{77e8bb28}_{16}, \\
&\phantom{= \{}\ \texttt{8d850302}_{16},\ \texttt{7a83f18c}_{16},\ \texttt{226c5eff}_{16},\ \texttt{f36a0313}_{16}, \\
&\phantom{= \{}\ \texttt{e9775cd9}_{16},\ \texttt{607d23c2}_{16},\ \texttt{fba9c48c}_{16},\ \texttt{bf7b3097}_{16}, \\
&\phantom{= \{}\ \texttt{2f612898}_{16},\ \texttt{61e29915}_{16},\ \texttt{dadecc5b}_{16},\ \texttt{2f533059}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
M'_5 &= \texttt{A4 74 25 81 8D C8 4F 86 73 6E 90 72 28 BB E8 77} \\
&\ \texttt{02 03 85 8D 8C F1 83 7A FF 5E 6C 22 13 03 6A F3} \\
&\ \texttt{D9 5C 77 E9 C2 23 7D 60 8C C4 A9 FB 97 30 8B BF} \\
&\ \texttt{98 28 61 2F 15 99 E2 61 5B CC DE DA 59 30 53 2F} \\
&= \{\ \texttt{812574a4}_{16},\ \texttt{864fc88d}_{16},\ \texttt{72906e73}_{16},\ \texttt{77e8bb28}_{16}, \\
&\phantom{= \{}\ \texttt{8d850302}_{16},\ \texttt{7a83f18c}_{16},\ \texttt{226c5eff}_{16},\ \texttt{f36a0313}_{16}, \\
&\phantom{= \{}\ \texttt{e9775cd9}_{16},\ \texttt{607d23c2}_{16},\ \texttt{fba9c48c}_{16},\ \texttt{bf8b3097}_{16}, \\
&\phantom{= \{}\ \texttt{2f612898}_{16},\ \texttt{61e29915}_{16},\ \texttt{dadecc5b}_{16},\ \texttt{2f533059}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
IHV_5 &= \texttt{E745A147086391F0910F3B97AE85BE73} \\
&= \{\texttt{47a145e7}_{16}, \texttt{f0916308}_{16}, \texttt{973b0f91}_{16}, \texttt{73be85ae}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
IHV'_5 &= \texttt{E745A14768C24DF4F16EF79A0EE57A77} \\
&= \{\texttt{47a145e7}_{16}, \texttt{f44dc268}_{16}, \texttt{9af76ef1}_{16}, \texttt{777ae50e}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
\delta IHV_5 &= \{0, \delta b_5, \delta b_5, \delta b_5\} \\
\delta b_5 &= -2^5 - 2^7 - 2^{13} + 2^{15} - 2^{18} - 2^{22} + 2^{26}
\end{aligned}
$$

D.3.2 Block 2 of 8

Table D-8: Differential Path - block 2

Using $\delta m_{11} = -2^{16}$

t	Bits Q_t: $b_{31} \dots b_0$	#
-3	`01000111 10100001 01000101 11100111`	32
-2	`01110+11 -+111-10 1++00101 -0-01110`	32
-1	`1001+-1- ++11-+11 0++0111- 1++10001`	32
0	`11110+00 -+0-++01 +1-0001- 0++01000`	32
1	`0.+.11.- ++1--+.. 0+1...1- .++.....`	18
2	`0.+0+0.1 -.+.-... 1+0..!+. ..-.....`	15
3	`+1+10+.+ 10+1-0.. 0++...11 .00.....`	20
4	`1.+.--.+ 1.+0.... .+1...-. ..+0.^..`	14
5	`1.11-10. -0.-0000 01-000-0 00110+00`	29
6	`110.0110 -11+1111 1-+111+1 11--1011`	31
7	`+0111-.. -..-.1.. --+.0.-. ..00.0..`	17
8	`001.+-.. +.!-.+.. +...+.-. ..-+....`	14
9	`+00.1-.. 0..+.1.. -10.0... ..-+...0`	15
10	`++100-0. 00.+.-00 -.000000 !1+.00.0`	26
11	`-0110+10 010-0-11 -001.111 ^111110-`	31
12	`1000+000 101+1-^0 -.+0.00^ ++10010+`	30
13	`----00-- ++---+-+ +^+11--- 0-0++++-`	32
14	`110--+++ ++++1000 +---1100 -+++++++`	32
15	`1110+100 +++0101+ +010+110 010111-0`	32
16	`..1+1101 1++1...1 1001-.0. 110000-0`	25
17	`!.1.1.1. 100..+.0 1...1.1. ..-..100`	16
18	`..-^..-. .^+..+.1 ....1.-. ..-...10`	12
19	`........ ..+..-.- ....-... .0+....+`	7
20	`.0^...^. ..0.1-.. .0....^. .1+..0..`	11
21	`.1....0. ..^.1-.^ .1..^... .+0..1.^`	12
22	`.+....1. ....-1.. .-...... ..^..-..`	7
23	`......-. 0...10.. ....0... .^......`	6
24	`.^.0.... 1...0... .^..1... 0....^..`	8
25	`......^. +....... ....+... 1.......`	4
26	`..0-.... .......0 ........ +.......`	4
27	`..1-.... ^......1 ....^... ........`	5
28	`..-+.... .......- ........ ^.......`	4
29	`...0.... .......0 ........ ........`	2
30	`..^1.... .......+ ........ ........`	3
31	`........ .......+ ........ ........`	1
32	`........ ........ ........ ........`	0
33	`........ .......! ........ ........`	1
$34-60$	`........ ........ ........ ........`	0
61	`........ ........ ........ ........`	
62	`.....-.. ........ ........ ........`	
63	`.....-.. ........ ........ ........`	
64	`.....-.. ........ ........ ........`	

Table D-9: Block 2 found using path in Table D-8

$$
\begin{aligned}
M_6 &= \texttt{B3 DD 11 72 78 E4 94 40 14 33 63 0E 74 61 C1 DC}\\
&\ \texttt{9B 80 1B 2E 55 20 15 A5 13 FF 7A E7 97 3E F4 4B}\\
&\ \texttt{83 52 E4 E0 49 79 B3 1E B6 00 65 4D 51 F4 A4 81}\\
&\ \texttt{CE BE 3F 0B D0 99 D1 30 D1 45 6F AB E0 4A 3E 98}\\
&= \{\ \texttt{7211ddb3}_{16},\ \texttt{4094e478}_{16},\ \texttt{0e633314}_{16},\ \texttt{dcc16174}_{16},\\
&\ \ \texttt{2e1b809b}_{16},\ \texttt{a5152055}_{16},\ \texttt{e77aff13}_{16},\ \texttt{4bf43e97}_{16},\\
&\ \ \texttt{e0e45283}_{16},\ \texttt{1eb37949}_{16},\ \texttt{4d6500b6}_{16},\ \texttt{81a4f451}_{16},\\
&\ \ \texttt{0b3fbece}_{16},\ \texttt{30d199d0}_{16},\ \texttt{ab6f45d1}_{16},\ \texttt{983e4ae0}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
M'_6 &= \texttt{B3 DD 11 72 78 E4 94 40 14 33 63 0E 74 61 C1 DC}\\
&\ \texttt{9B 80 1B 2E 55 20 15 A5 13 FF 7A E7 97 3E F4 4B}\\
&\ \texttt{83 52 E4 E0 49 79 B3 1E B6 00 65 4D 51 F4 A3 81}\\
&\ \texttt{CE BE 3F 0B D0 99 D1 30 D1 45 6F AB E0 4A 3E 98}\\
&= \{\ \texttt{7211ddb3}_{16},\ \texttt{4094e478}_{16},\ \texttt{0e633314}_{16},\ \texttt{dcc16174}_{16},\\
&\ \ \texttt{2e1b809b}_{16},\ \texttt{a5152055}_{16},\ \texttt{e77aff13}_{16},\ \texttt{4bf43e97}_{16},\\
&\ \ \texttt{e0e45283}_{16},\ \texttt{1eb37949}_{16},\ \texttt{4d6500b6}_{16},\ \texttt{81a3f451}_{16},\\
&\ \ \texttt{0b3fbece}_{16},\ \texttt{30d199d0}_{16},\ \texttt{ab6f45d1}_{16},\ \texttt{983e4ae0}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
IHV_6 &= \texttt{6900F0DD0821F13B2AF6DF5D3521BFC7}\\
&= \{\texttt{ddf00069}_{16}, \texttt{3bf12108}_{16}, \texttt{5ddff62a}_{16}, \texttt{c7bf2135}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
IHV'_6 &= \texttt{6900F0DD6880AD3B8A559C5D95807BC7}\\
&= \{\texttt{ddf00069}_{16}, \texttt{3bad8068}_{16}, \texttt{5d9c558a}_{16}, \texttt{c77b8095}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
\delta IHV_6 &= \{0, \delta b_6, \delta b_6, \delta b_6\}\\
\delta b_6 &= -2^5 - 2^7 - 2^{13} + 2^{15} - 2^{18} - 2^{22}
\end{aligned}
$$

D.3.3 Block 3 of 8

Table D-10: Differential Path - block 3

Using $\delta m_{11} = +2^{12}$

t	Bits Q_t: $b_{31} \dots b_0$	#
-3	11011101 11110000 00000000 01101001	32
-2	11000111 -+111-11 +0-0000- +0-10101	32
-1	01011101 1-0111-- -1-101-+ +0-01010	32
0	00111011 1-1-++01 +0-0000- 0++01000	32
1	.......- .+.-01-- 1---..10 --+.1...	17
2	0..!...0 .+..0-0- -1.+0..- +0-.-...	17
3	01.....1 .-.0-1+. +00+0..- -+1.0...	18
4	+0.^.... .-..1--1 +...+... 0-+.1...	14
5	110+000^ 0.00+1+. 0001-001 00-0.010	29
6	+110110+ 111110-0 -1111111 00-10111	32
7	-..+..-0 ....00-1 +..01... -.-...-.	14
8	-1!1..-0 .....1-. +..-.... 1..^..0.	13
9	10.1..+0 ..0..+-. ...+.0.. 1!1+0.0.	16
10	0.0.00+0 .01.00.0 100-0100 ..011..^	23
11	1110111+ 01-01-01 011+1-11 0101-01+	32
12	001.110- 10110-11 01001+.0 11+0-110	30
13	-++^+++1 -+++++-+ -----+^1 +--10--1	32
14	010+1100 111+---- +-0+11-+ +++++++.	31
15	11+0110+ +000-111 010--011 -110-01.	31
16	..+00.0- ..01-001 00.0..01 +100110.	23
17	..00..11 ^.1.0.0. .-.0^... 1.1.1.+.	15
18	..10..+^ ..+.1.1. .-.1.... 1.+...+.	12
19	...-.... ......-. .+.+.... +....0-.	7
20	.0...0^. ..^...0. 1+...0.. ..^..1-.	11
21	.1.^.1.. ..0...1. 1+.^.1.. ^....-0.	12
22	.+...-.. ..1..... +1...+.. ......^.	7
23	....+.0... 10...... 0....^..	6
24	.^...^.0 ....1... 0....^.. 1...0...	8
25	....^.-... .... -...1...	4
26	......0+ ....0.... ....-...	4
27	......1+ ....^... ...1.... ^.......	5
28	......+- ....+.... ....^...	4
29	.......0 ....0....	2
30	......^1 ....-....	3
31	....-....	1
32	....	0
33	....!....	1
$34-60$	....	0
61	....	
62	....+......	
63	....+......	
64	....+......	

Table D-11: Block 3 found using path in Table D-10

$$
\begin{aligned}
M_7 &= \texttt{85 C8 C4 FB 29 7B 86 B5 77 52 CD 64 19 80 9F E3}\\
&\ \texttt{7E 62 86 F0 77 32 D1 E0 69 A5 B4 E5 66 70 B8 BB}\\
&\ \texttt{BA E5 C2 11 74 2A 13 1D 05 71 1C F1 FE 22 AF 93}\\
&\ \texttt{3F 1E EF 22 47 62 E3 AA DA C1 7C 40 E4 48 CA 41}\\
&= \{\ \texttt{fbc4c885}_{16},\ \texttt{b5867b29}_{16},\ \texttt{64cd5277}_{16},\ \texttt{e39f8019}_{16},\\
&\ \ \texttt{f086627e}_{16},\ \texttt{e0d13277}_{16},\ \texttt{e5b4a569}_{16},\ \texttt{bbb87066}_{16},\\
&\ \ \texttt{11c2e5ba}_{16},\ \texttt{1d132a74}_{16},\ \texttt{f11c7105}_{16},\ \texttt{93af22fe}_{16},\\
&\ \ \texttt{22ef1e3f}_{16},\ \texttt{aae36247}_{16},\ \texttt{407cc1da}_{16},\ \texttt{41ca48e4}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
M'_7 &= \texttt{85 C8 C4 FB 29 7B 86 B5 77 52 CD 64 19 80 9F E3}\\
&\ \texttt{7E 62 86 F0 77 32 D1 E0 69 A5 B4 E5 66 70 B8 BB}\\
&\ \texttt{BA E5 C2 11 74 2A 13 1D 05 71 1C F1 FE 32 AF 93}\\
&\ \texttt{3F 1E EF 22 47 62 E3 AA DA C1 7C 40 E4 48 CA 41}\\
&= \{\ \texttt{fbc4c885}_{16},\ \texttt{b5867b29}_{16},\ \texttt{64cd5277}_{16},\ \texttt{e39f8019}_{16},\\
&\ \ \texttt{f086627e}_{16},\ \texttt{e0d13277}_{16},\ \texttt{e5b4a569}_{16},\ \texttt{bbb87066}_{16},\\
&\ \ \texttt{11c2e5ba}_{16},\ \texttt{1d132a74}_{16},\ \texttt{f11c7105}_{16},\ \texttt{93af32fe}_{16},\\
&\ \ \texttt{22ef1e3f}_{16},\ \texttt{aae36247}_{16},\ \texttt{407cc1da}_{16},\ \texttt{41ca48e4}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
IHV_7 &= \texttt{6F48D9E5383E55D0FC43ED4D20ABF6F8}\\
&= \{\texttt{e5d9486f}_{16}, \texttt{d0553e38}_{16}, \texttt{4ded43fc}_{16}, \texttt{f8f6ab20}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
IHV'_7 &= \texttt{6F48D9E5989D51D05CA3E94D800AF3F8}\\
&= \{\texttt{e5d9486f}_{16}, \texttt{d0519d98}_{16}, \texttt{4de9a35c}_{16}, \texttt{f8f30a80}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
\delta IHV_7 &= \{0, \delta b_7, \delta b_7, \delta b_7\}\\
\delta b_7 &= -2^5 - 2^7 - 2^{13} + 2^{15} - 2^{18}
\end{aligned}
$$

D.3.4 Block 4 of 8

Table D-12: Differential Path - block 4

Using $\delta m_{11} = +2^8$

t	Bits Q_t: $b_{31} \dots b_0$	#
-3	`11100101 11011001 01001000 01101111`	32
-2	`11111000 11110-1+ -0-0101- +0-00000`	32
-1	`01001101 11101-01 +-+00011 -1-11100`	32
0	`11010000 01010-01 +0-111-+ +0-11000`	32
1	`.0..0.-0 ....-+.. .+1+..01 -1-...0.`	15
2	`..!.+.0- ....10.. 0+0+..00 --+0..0.`	17
3	`!...0.++ .0.010.. .+.+...- +-10..+.`	16
4	`....11++ .1!0...^ .-.+...1 -.0-..-.`	15
5	`0010.0.+ 0+.+00.- 010-0000 -0.-00-0`	27
6	`11110+1- 0+0-1101 101-1111 .10+11+1`	31
7	`10+..1.0 +..+..00 ..1-.... 10.+..1.`	15
8	`110..-.0 +1.+..+1 ..0...1. .-....0.`	14
9	`-1-..+.. 1.....+. ..-01.01 .-.0.^..`	14
10	`-010.+0. 11^0...0 .01.00+0 0-..0-00`	23
11	`00+10110 -1+10001 011001++ 1.011011`	31
12	`+0101100 1-100110 01+0+111 1011011.`	31
13	`-1-1---- 0--++--1 --++-100 00+----.`	31
14	`0-++10+- -+++++++ +1-11-++ ++0++10.`	31
15	`1-110010 00-.-001 011-10-1 -01101-.`	30
16	`0110..+1 011^111+ 1.1+.0+1 10+00.1.`	26
17	`.1.....+ ..1.1... ...+..1. 1.+1..1.`	10
18	`..0...^1 ..-....^ ...1..0. ..01..+.`	10
19	`.00....- ......1. ...-.0.. ..^-....`	8
20	`.1+..0.. ..^..... ....01.. .0....^.`	8
21	`0+1..1.^ ......+. ...^1+.. .1.^....`	11
22	`..0..+.. ......+. ....+... .+......`	5
23	`+^...... ......-. 0...1^.. ....0...`	7
24	`+....^.. ...0..0. 1...0... .^..1...`	8
25	`-....... ......1. -....... ....-...`	4
26	`-....... ..0+.... .......0 ........`	4
27	`1....... ..1+.... ^......1 ....^...`	6
28	`0....... ..+-.... .......+ ........`	4
29	`........ ...0.... .......0 ........`	2
30	`........ ..^1.... .......- ........`	3
31	`........ ........ .......- ........`	1
32	`........ ........ ........ ........`	0
33	`........ ........ .......! ........`	1
$34-60$	`........ ........ ........ ........`	0
61	`........ ........ ........ ........`	
62	`........ .....+.. ........ ........`	
63	`........ .....+.. ........ ........`	
64	`........ .....+.. ........ ........`	

Table D-13: Block 4 found using path in Table D-12

$$
\begin{aligned}
M_8 &= \texttt{A8 79 A0 3D 3C F6 65 F2 39 C7 F3 FE 82 B3 84 E8}\\
&\quad\ \texttt{35 E7 C9 E8 BD EE 30 C2 68 A2 12 12 84 78 9D F4}\\
&\quad\ \texttt{2F 44 90 6F 19 B7 90 26 46 44 36 E1 DA 64 FA 0C}\\
&\quad\ \texttt{53 A3 77 FA 0D 2B 01 2B 7D DC 28 55 DA E5 B5 51}\\
&= \{\ \texttt{3da079a8}_{16},\ \texttt{f265f63c}_{16},\ \texttt{fef3c739}_{16},\ \texttt{e884b382}_{16},\\
&\qquad \texttt{e8c9e735}_{16},\ \texttt{c230eebd}_{16},\ \texttt{1212a268}_{16},\ \texttt{f49d7884}_{16},\\
&\qquad \texttt{6f90442f}_{16},\ \texttt{2690b719}_{16},\ \texttt{e1364446}_{16},\ \texttt{0cfa64da}_{16},\\
&\qquad \texttt{fa77a353}_{16},\ \texttt{2b012b0d}_{16},\ \texttt{5528dc7d}_{16},\ \texttt{51b5e5da}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
M'_8 &= \texttt{A8 79 A0 3D 3C F6 65 F2 39 C7 F3 FE 82 B3 84 E8}\\
&\quad\ \texttt{35 E7 C9 E8 BD EE 30 C2 68 A2 12 12 84 78 9D F4}\\
&\quad\ \texttt{2F 44 90 6F 19 B7 90 26 46 44 36 E1 DA 65 FA 0C}\\
&\quad\ \texttt{53 A3 77 FA 0D 2B 01 2B 7D DC 28 55 DA E5 B5 51}\\
&= \{\ \texttt{3da079a8}_{16},\ \texttt{f265f63c}_{16},\ \texttt{fef3c739}_{16},\ \texttt{e884b382}_{16},\\
&\qquad \texttt{e8c9e735}_{16},\ \texttt{c230eebd}_{16},\ \texttt{1212a268}_{16},\ \texttt{f49d7884}_{16},\\
&\qquad \texttt{6f90442f}_{16},\ \texttt{2690b719}_{16},\ \texttt{e1364446}_{16},\ \texttt{0cfa65da}_{16},\\
&\qquad \texttt{fa77a353}_{16},\ \texttt{2b012b0d}_{16},\ \texttt{5528dc7d}_{16},\ \texttt{51b5e5da}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
IHV_8 &= \texttt{80D9AE060626A79399F4E05A0E7F318F}\\
&= \{\texttt{06aed980}_{16}, \texttt{93a72606}_{16}, \texttt{5ae0f499}_{16}, \texttt{8f317f0e}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
IHV'_8 &= \texttt{80D9AE066685A793F953E15A6EDE318F}\\
&= \{\texttt{06aed980}_{16}, \texttt{93a78566}_{16}, \texttt{5ae153f9}_{16}, \texttt{8f31de6e}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
\delta IHV_8 &= \{0, \delta b_8, \delta b_8, \delta b_8\}\\
\delta b_8 &= -2^5 - 2^7 - 2^{13} + 2^{15}
\end{aligned}
$$

D.3.5 Block 5 of 8

Table D-14: Differential Path - block 5

Using $\delta m_{11} = -2^5$

t	Bits Q_t: $b_{31} \dots b_0$	#
-3	00000110 10101110 11011001 10000000	32
-2	10001111 00110001 +1-1111- 0++01110	32
-1	01011010 1110000+ -1-10-++ 1++11001	32
0	10010011 10100111 +0-001-+ 0++00110	32
1	0....0.. +..-.1.1 -.0+.1-+ 1++.....	15
2	00.!.+.. -..0.-.. -.+-.... -0-.....	13
3	+1..^1.. 1..1.-.. -.1+..00 +-+.....	15
4	00..+1^. 0....-.! +.1-.... +-+.....	14
5	10000.-0 -0000+0. 10.00000 0++00.00	28
6	+1111001 -1111110 01001111 +0+11011	32
7	+.....10 .....00. ..0..0.- +1+..00.	13
8	-00....1 0..!..+. .^-....0 .-0..-0.	14
9	110....+ .0....+. .+0....0 011..--.	14
10	0.+0100+ 0000^0.0 0010...0 .1.!01+0	25
11	10+10111 1+11-101 11.10000 00001++1	31
12	0000+00- .++10101 0+00101+ 000101-1	31
13	-+011--- .--+---- --10+--- +----+1-	31
14	10---1-- .0111011 -+++++++ --------	31
15	11000101 .+01110- 010+000+ 0-..-1+0	29
16	0-001.10 .0.0.1.. 111+1111 10^^100-	26
17	.-1..-.. .11..1.^ ...0...+ .0.!-.--	14
18	^+...-.. ..0..-.. ...1...0 ....-..0	9
19	.0..0+.. ..+..... ....- 0...+.^-	9
20	.0..1+.. .....^.. ....+ 1...+..0	8
21	....+0.. 0.^..... .0.....1 +...1..1	9
22	...0.0.. 1....... .1.....- ....-...	6
23	....^... -....... .-.0.... ^......0	6
24	..0+.... 0.....0. ...1...^ ..0.^..1	9
25	..1+.... 1....... .^.+.... ..1....+	7
26	..+-.... .....0-. ....+.....	5
27	...0.... .....1-. ...^.... ..0....^	6
28	..^1.... .....-+. ....-.....	5
29	....0. ....0.....	2
30	....^1. ....+.....	3
31	....+.....	1
32	....	0
33	....!.....	1
$34-60$	....	0
61	....	
62	.... -.......	
63	.... -.......	
64	.... -.......	

Table D-15: Block 5 found using path in Table D-14

$$
\begin{aligned}
M_9 &= \texttt{51 E2 80 34 11 21 20 B5 E7 9E C5 F2 6A 9F 69 DA} \\
&\ \texttt{85 D7 4E F6 A9 7A 0B 11 64 EF A2 5F B1 AE 26 BA} \\
&\ \texttt{45 1C CD A7 A2 E7 84 33 9C 44 7D 56 25 49 A6 0B} \\
&\ \texttt{F0 67 62 94 BF 58 0C 91 9E C4 57 02 5D 3C 78 60} \\
&= \{\ \texttt{3480e251}_{16},\ \texttt{b5202111}_{16},\ \texttt{f2c59ee7}_{16},\ \texttt{da699f6a}_{16}, \\
&\ \ \texttt{f64ed785}_{16},\ \texttt{110b7aa9}_{16},\ \texttt{5fa2ef64}_{16},\ \texttt{ba26aeb1}_{16}, \\
&\ \ \texttt{a7cd1c45}_{16},\ \texttt{3384e7a2}_{16},\ \texttt{567d449c}_{16},\ \texttt{0ba64925}_{16}, \\
&\ \ \texttt{946267f0}_{16},\ \texttt{910c58bf}_{16},\ \texttt{0257c49e}_{16},\ \texttt{60783c5d}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
M'_9 &= \texttt{51 E2 80 34 11 21 20 B5 E7 9E C5 F2 6A 9F 69 DA} \\
&\ \texttt{85 D7 4E F6 A9 7A 0B 11 64 EF A2 5F B1 AE 26 BA} \\
&\ \texttt{45 1C CD A7 A2 E7 84 33 9C 44 7D 56 05 49 A6 0B} \\
&\ \texttt{F0 67 62 94 BF 58 0C 91 9E C4 57 02 5D 3C 78 60} \\
&= \{\ \texttt{3480e251}_{16},\ \texttt{b5202111}_{16},\ \texttt{f2c59ee7}_{16},\ \texttt{da699f6a}_{16}, \\
&\ \ \texttt{f64ed785}_{16},\ \texttt{110b7aa9}_{16},\ \texttt{5fa2ef64}_{16},\ \texttt{ba26aeb1}_{16}, \\
&\ \ \texttt{a7cd1c45}_{16},\ \texttt{3384e7a2}_{16},\ \texttt{567d449c}_{16},\ \texttt{0ba64905}_{16}, \\
&\ \ \texttt{946267f0}_{16},\ \texttt{910c58bf}_{16},\ \texttt{0257c49e}_{16},\ \texttt{60783c5d}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
IHV_9 &= \texttt{73A70AC09AC9B2233ECC7BE4C30C6488} \\
&= \{\texttt{c00aa773}_{16}, \texttt{23b2c99a}_{16}, \texttt{e47bcc3e}_{16}, \texttt{88640cc3}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
IHV'_9 &= \texttt{73A70AC0FAA8B2239EAB7BE423EC6388} \\
&= \{\texttt{c00aa773}_{16}, \texttt{23b2a8fa}_{16}, \texttt{e47bab9e}_{16}, \texttt{8863ec23}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
\delta IHV_9 &= \{0, \delta b_9, \delta b_9, \delta b_9\} \\
\delta b_9 &= -2^5 - 2^7 - 2^{13}
\end{aligned}
$$

D.3.6 Block 6 of 8

Table D-16: Differential Path - block 6

Using $\delta m_{11} = +2^3$

t	Bits Q_t: $b_{31} \dots b_0$	#
-3	11000000 00001010 10100111 01110011	32
-2	10001000 01100-++ +++01100 --+00011	32
-1	11100100 01111011 1-+01-++ +0-11110	32
0	00100011 10110010 1-+0100- 1++11010	32
1	.1..0-+. +..+..0. .....00- 0++.10..	15
2	00..-0+. +..-..+. .11..1.. .-1.+0..	15
3	1-1.-1.1 ...-..+. ^.1..0.0 .1-.0+..	16
4	1+....11 0.!..... +^+..... .++.1-1.	14
5	0-00101+ 10.00^00 0+000000 0++0-100	31
6	1011110+ 01011-11 10101111 1.0100-1	31
7	101...+0 -....0.. .1.-.... .1-.11+.	14
8	0.+...10 +.!..0.. .-.+.... ..00+0+.	14
9	0.+.!.1- -.0..... .0.1...0 ..-.++..	13
10	10-0.001 -00.^000 0-.0.001 ..01.-00	25
11	11-10100 -1-0-111 1000011+ 00010-11	32
12	00+000-1 10-11111 1-00000+ 101-0100	32
13	0000++-1 0-0+++++ +1+++0-- -+++0-++	32
14	+------+ -----111 0+------ 1------0	32
15	1111111- 1+101011 00011-.0 110-1010	31
16	+01-0101 0100+..0 .00011^- .011001.	27
17	...0...+ .1..0..1 .....1.. ...-.--.	9
18	^..+...+ ....0..- ...0...^ 0..-....	9
19	...+..01 ....-... .... 10.1.^^.	9
20	...0.01- ..0....^ ...+.... +1.0..0.	11
21	...1..-. ..1.^... ...+.0.. 1-....1.	9
22	.....-0^ ..+..... 0..-.... 0.....+.	8
23	.0...-1. .... 1..-.+.. .^......	7
24	.1...+.. ..^..... -..1.+.. ......^.	7
25	.-...+.. .... 0..0.-..	5
26	.....+.. .... 0....0.. ....0...	4
27	.^...1.. .... 1....1.. ...01...	6
28	.....0.. .... +....... ...1-...	4
29	....-1...	2
30	.... ^....... ...1-...	3
31	....-+...	2
32	....0....	0
33	....!!...	2
$34-60$	....	0
61	....	
62	....+.....	
63	....+.....	
64	....+.....	

Table D-17: Block 6 found using path in Table D-16

$$
\begin{aligned}
M_{10} &= \texttt{B9 82 96 C0 AB 9F E5 B1 D3 53 88 2E 26 C1 F7 21} \\
&\ \texttt{B4 18 99 D9 72 B5 A1 D5 05 0B 68 45 36 44 80 10} \\
&\ \texttt{AF 8C 7A FF 7C E8 EA CC B9 B1 FB BD C9 29 D4 F5} \\
&\ \texttt{D4 99 FB 81 29 24 DF 30 2C B3 C4 50 23 38 62 97} \\
&= \{\ \texttt{c09682b9}_{16},\ \texttt{b1e59fab}_{16},\ \texttt{2e8853d3}_{16},\ \texttt{21f7c126}_{16}, \\
&\ \ \texttt{d99918b4}_{16},\ \texttt{d5a1b572}_{16},\ \texttt{45680b05}_{16},\ \texttt{10804436}_{16}, \\
&\ \ \texttt{ff7a8caf}_{16},\ \texttt{cceae87c}_{16},\ \texttt{bdfbb1b9}_{16},\ \texttt{f5d429c9}_{16}, \\
&\ \ \texttt{81fb99d4}_{16},\ \texttt{30df2429}_{16},\ \texttt{50c4b32c}_{16},\ \texttt{97623823}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
M'_{10} &= \texttt{B9 82 96 C0 AB 9F E5 B1 D3 53 88 2E 26 C1 F7 21} \\
&\ \texttt{B4 18 99 D9 72 B5 A1 D5 05 0B 68 45 36 44 80 10} \\
&\ \texttt{AF 8C 7A FF 7C E8 EA CC B9 B1 FB BD D1 29 D4 F5} \\
&\ \texttt{D4 99 FB 81 29 24 DF 30 2C B3 C4 50 23 38 62 97} \\
&= \{\ \texttt{c09682b9}_{16},\ \texttt{b1e59fab}_{16},\ \texttt{2e8853d3}_{16},\ \texttt{21f7c126}_{16}, \\
&\ \ \texttt{d99918b4}_{16},\ \texttt{d5a1b572}_{16},\ \texttt{45680b05}_{16},\ \texttt{10804436}_{16}, \\
&\ \ \texttt{ff7a8caf}_{16},\ \texttt{cceae87c}_{16},\ \texttt{bdfbb1b9}_{16},\ \texttt{f5d429d1}_{16}, \\
&\ \ \texttt{81fb99d4}_{16},\ \texttt{30df2429}_{16},\ \texttt{50c4b32c}_{16},\ \texttt{97623823}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
IHV_{10} &= \texttt{DE56FC8A3A0A1FEBBE6E537DB6629AC4} \\
&= \{\texttt{8afc56de}_{16}, \texttt{eb1f0a3a}_{16}, \texttt{7d536ebe}_{16}, \texttt{c49a62b6}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
IHV'_{10} &= \texttt{DE56FC8A9A091FEB1E6E537D16629AC4} \\
&= \{\texttt{8afc56de}_{16}, \texttt{eb1f099a}_{16}, \texttt{7d536e1e}_{16}, \texttt{c49a6216}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
\delta IHV_{10} &= \{0, \delta b_{10}, \delta b_{10}, \delta b_{10}\} \\
\delta b_{10} &= -2^5 - 2^7
\end{aligned}
$$

D.3.7 Block 7 of 8

Table D-18: Differential Path - block 7

Using $\delta m_{11} = +2^{29}$

t	Bits Q_t: $b_{31} \dots b_0$	#
-3	10001010 11111100 01010110 11011110	32
-2	11000100 10011010 01100010 -0-10110	32
-1	01111101 01010011 01101110 -0-11110	32
0	11101011 00011111 000010-+ +0-11010	32
1	..0..0.. ....11.. 0-0-..01 -1-.....	13
2	.1.!0+.. ....1+.. -0++..00 --+.....	15
3	.1!.01.. .0..+-.. 1-0-.... --+.....	14
4	!-..-1.. .0..+-!. +1++.... .+......	13
5	!-00-.00 ^-001-.0 101+0000 1+000000	30
6	!+11-011 ++11--01 1.+-1111 1.111111	30
7	!1..-... 00.^--.! -.01.... .1^.^...	15
8	!1..+... 10!-.... -.0-.0.. ..+0+..0	15
9	..!.1... ...010.. -..+.0.. .!001^.0	14
10	00.!-010 00.1..10 .00+!+.0 .01+1-1-	25
11	110.-111 1100^011 01110+01 001-000+	31
12	.11^00+1 0010+1+^ 00^1111. 1-0-0+-0	30
13	^1+----0 1-0+0+0- +++++++1 +--+-++0	32
14	--1110-+ +++++0+1 00000010 +--0---.	31
15	1+1+1-1- 011-1+10 0000000- 011-.10.	30
16	01...00+ 10111+1. ..+..1.. 100-^01.	21
17	.0.^.+.1 .1.^.+.. ..-..1.^ .0.0..0.	13
18	.......1 .+...+.. ..1..+.. .1.1..1.	8
19	0....^.+ ....0-.. ..-..... .-....-.	8
20	1...0... .^..1-.. 0....^.. .1.....0	9
21	+...1..^ ...0-0.. 1.^..... .0....^0	11
22	....+... ...1.^.. +....... .1.....+	6
23	^......0 ..0-^... 1....... .+.0....	8
24	....^..1 ..10.... 0.....0. ...1...^	8
25	.......- ..-+.... ....^.-....	5
26	..0..... ...-.... .....0+.	4
27	..1....^ ..^1.... .....1+. ...^....	7
28	..+..... ...0.... .....+-.	4
29	..0..... ....0.	2
30	..-..... ....^1.	3
31	..-.....	1
32	....	0
33	..!.....	1
$34-60$	....	0
61	....	
62	.... +.......	
63	.... +.......	
64	.... +.......	

Table D-19: Block 7 found using path in Table D-18

$$
\begin{aligned}
M_{11} &= \texttt{93 96 B3 A4 6C D0 FF 7F 14 26 71 1C 45 92 97 B6} \\
&\ \texttt{5D 1C EF 66 C1 87 51 E0 94 BF 08 F3 B2 98 1C 5C} \\
&\ \texttt{CE 52 D9 63 D5 A4 25 9A 64 55 7E 4D 1B 9E FE 0D} \\
&\ \texttt{9A 51 6D 1E 6E C8 BB 37 06 68 25 AE A6 36 16 60} \\
&= \{\ \texttt{a4b39693}_{16},\ \texttt{7fffd06c}_{16},\ \texttt{1c712614}_{16},\ \texttt{b6979245}_{16}, \\
&\ \ \texttt{66ef1c5d}_{16},\ \texttt{e05187c1}_{16},\ \texttt{f308bf94}_{16},\ \texttt{5c1c98b2}_{16}, \\
&\ \ \texttt{63d952ce}_{16},\ \texttt{9a25a4d5}_{16},\ \texttt{4d7e5564}_{16},\ \texttt{0dfe9e1b}_{16}, \\
&\ \ \texttt{1e6d519a}_{16},\ \texttt{37bbc86e}_{16},\ \texttt{ae256806}_{16},\ \texttt{601636a6}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
M'_{11} &= \texttt{93 96 B3 A4 6C D0 FF 7F 14 26 71 1C 45 92 97 B6} \\
&\ \texttt{5D 1C EF 66 C1 87 51 E0 94 BF 08 F3 B2 98 1C 5C} \\
&\ \texttt{CE 52 D9 63 D5 A4 25 9A 64 55 7E 4D 1B 9E FE 2D} \\
&\ \texttt{9A 51 6D 1E 6E C8 BB 37 06 68 25 AE A6 36 16 60} \\
&= \{\ \texttt{a4b39693}_{16},\ \texttt{7fffd06c}_{16},\ \texttt{1c712614}_{16},\ \texttt{b6979245}_{16}, \\
&\ \ \texttt{66ef1c5d}_{16},\ \texttt{e05187c1}_{16},\ \texttt{f308bf94}_{16},\ \texttt{5c1c98b2}_{16}, \\
&\ \ \texttt{63d952ce}_{16},\ \texttt{9a25a4d5}_{16},\ \texttt{4d7e5564}_{16},\ \texttt{2dfe9e1b}_{16}, \\
&\ \ \texttt{1e6d519a}_{16},\ \texttt{37bbc86e}_{16},\ \texttt{ae256806}_{16},\ \texttt{601636a6}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
IHV_{11} &= \texttt{DCA82596835B2D4F2EDB818BFEE0D521} \\
&= \{\texttt{9625a8dc}_{16}, \texttt{4f2d5b83}_{16}, \texttt{8b81db2e}_{16}, \texttt{21d5e0fe}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
IHV'_{11} &= \texttt{DCA82596635B2D4F0EDB818BDEE0D521} \\
&= \{\texttt{9625a8dc}_{16}, \texttt{4f2d5b63}_{16}, \texttt{8b81db0e}_{16}, \texttt{21d5e0de}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
\delta IHV_{11} &= \{0, \delta b_{11}, \delta b_{11}, \delta b_{11}\} \\
\delta b_{11} &= -2^5
\end{aligned}
$$

D.3.8 Block 8 of 8

Table D-20: Differential Path - block 8

Using $\delta m_{11} = +2^{27}$

t	Bits Q_t: $b_{31} \dots b_0$	#
-3	10010110 00100101 10101000 11011100	32
-2	00100001 11010101 11100000 11-11110	32
-1	10001011 10000001 11011011 00-01110	32
0	01001111 00101101 01011011 -++00011	32
1	..1..... ......0. ..+-.... 0-+.....	7
2	..00.... ....0^-. ..1+.... 1-+.....	10
3	!.+0..1. ....0-+. ..1+.... .0-.....	11
4	.!0+..0. ....-1-. ...+.... .-+.....	10
5	..+-..-. ....+1-. .1...... .1-.1...	10
6	!.0-..0. ....1... .0.1.!.1 .10.00..	13
7	..1+..01 ....0.0. !+.....0 ..0.+1.1	13
8	!..0..-1 ....! .-.^...- ....-+.0	11
9	.!.0..0+ ......0. .+1-...+ 1-.-	12
10	......1+ .1....1. .100...+ 0.0.0-.-	13
11	.1...11+ .00101-1 !1+0.1.+ 0.101-10	24
12	00^0000- .-101111 .0-000.1 +^-10001	29
13	0+-00-+1 ^0--++-- ^-1+1-.- ++++----	31
14	+110+--- ---+0+-- -----100 .1110100	31
15	101-1-11 101010.0 1+1001.1 11110-0-	30
16	10010010 +00-.1^1 00101+.0 ....-	23
17	01.-.0.. ...0...+ .0..0..1 .....^.1	11
18	1+.-.... ^..+...+ ....0^.- ...0...0	11
19	+0.1.... ...+..01 ....-...	7
20	.-.0..0. ...0.01- ..0....^ ...+....	10
21	^-....1. ...1..-. ..1.^... ...+.0..	9
22	.0....+. .....-0^ ..+..... 0..-....	8
23	.1...... .0...-1. .... 1..-.+..	7
24	......^. .1...+.. ..^..... -..1.+..	7
25	....-...+.. .... 0..0.-..	5
26	....0... .....+.. .... 0....0..	4
27	...01... .^...1.. .... 1....1..	6
28	...1-... .....0.. .... +.......	4
29	...-1...	2
30	...1-... .... ^.......	3
31	...-+...	2
32	...0....	1
33	...!!...	2
$34-60$	....	0
61	....	
62	....+.....	
63	....+.....	
64	....+.....	

Table D-21: Block 8 found using path in Table D-20

$$
\begin{aligned}
M_{12} &= \texttt{2B D7 D1 16 25 A0 6A 90 73 9B 4D 0A 06 EA 87 2A} \\
& \texttt{3A F9 EB A1 26 29 BE D6 79 40 56 1B D9 37 4A 89} \\
& \texttt{D6 0F 0D 72 2C 9F EB 68 33 EC 53 F0 B0 FD 76 A2} \\
& \texttt{04 7B 66 C9 0F CE B1 D2 E2 2C C0 99 B9 A4 B9 3E} \\
&= \{\ \texttt{16d1d72b}_{16},\ \texttt{906aa025}_{16},\ \texttt{0a4d9b73}_{16},\ \texttt{2a87ea06}_{16}, \\
&\ \texttt{a1ebf93a}_{16},\ \texttt{d6be2926}_{16},\ \texttt{1b564079}_{16},\ \texttt{894a37d9}_{16}, \\
&\ \texttt{720d0fd6}_{16},\ \texttt{68eb9f2c}_{16},\ \texttt{f053ec33}_{16},\ \texttt{a276fdb0}_{16}, \\
&\ \texttt{c9667b04}_{16},\ \texttt{d2b1ce0f}_{16},\ \texttt{99c02ce2}_{16},\ \texttt{3eb9a4b9}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
M'_{12} &= \texttt{2B D7 D1 16 25 A0 6A 90 73 9B 4D 0A 06 EA 87 2A} \\
& \texttt{3A F9 EB A1 26 29 BE D6 79 40 56 1B D9 37 4A 89} \\
& \texttt{D6 0F 0D 72 2C 9F EB 68 33 EC 53 F0 B0 FD 76 AA} \\
& \texttt{04 7B 66 C9 0F CE B1 D2 E2 2C C0 99 B9 A4 B9 3E} \\
&= \{\ \texttt{16d1d72b}_{16},\ \texttt{906aa025}_{16},\ \texttt{0a4d9b73}_{16},\ \texttt{2a87ea06}_{16}, \\
&\ \texttt{a1ebf93a}_{16},\ \texttt{d6be2926}_{16},\ \texttt{1b564079}_{16},\ \texttt{894a37d9}_{16}, \\
&\ \texttt{720d0fd6}_{16},\ \texttt{68eb9f2c}_{16},\ \texttt{f053ec33}_{16},\ \texttt{aa76fdb0}_{16}, \\
&\ \texttt{c9667b04}_{16},\ \texttt{d2b1ce0f}_{16},\ \texttt{99c02ce2}_{16},\ \texttt{3eb9a4b9}_{16}\ \}
\end{aligned}
$$

$$
\begin{aligned}
IHV_{12} &= \texttt{505D9746FAB00B328018DBC34A87DF11} \\
&= \{\texttt{46975d50}_{16}, \texttt{320bb0fa}_{16}, \texttt{c3db1880}_{16}, \texttt{11df874a}_{16}\}
\end{aligned}
$$

$$
\begin{aligned}
IHV'_{12} &= \texttt{505D9746FAB00B328018DBC34A87DF11} \\
&= \{\texttt{46975d50}_{16}, \texttt{320bb0fa}_{16}, \texttt{c3db1880}_{16}, \texttt{11df874a}_{16}\}
\end{aligned}
$$

$$
\delta IHV_{12} = \{0, 0, 0, 0\}
$$

D.4 RSA Moduli

Table D-22: Upper Partial RSA Modulus 1

$$S_b \| S_c = X\|Y\|Z\|M_5\|M_6\|M_7\|M_8\|M_9\|M_{10}\|M_{11}\|M_{12}$$

=
```
1A09B4CB 40C7267A AF017F9B A4742581 8DC84F86 736E9072 28BBE877 0203858D
8CF1837A FF5E6C22 13036AF3 D95C77E9 C2237D60 8CC4A9FB 97307BBF 9828612F
1599E261 5BCCDEDA 5930532F B3DD1172 78E49440 1433630E 7461C1DC 9B801B2E
552015A5 13FF7AE7 973EF44B 8352E4E0 4979B31E B600654D 51F4A481 CEBE3F0B
D099D130 D1456FAB E04A3E98 85C8C4FB 297B86B5 7752CD64 19809FE3 7E6286F0
7732D1E0 69A5B4E5 6670B8BB BAE5C211 742A131D 05711CF1 FE22AF93 3F1EEF22
4762E3AA DAC17C40 E448CA41 A879A03D 3CF665F2 39C7F3FE 82B384E8 35E7C9E8
BDEE30C2 68A21212 84789DF4 2F44906F 19B79026 464436E1 DA64FA0C 53A377FA
0D2B012B 7DDC2855 DAE5B551 51E28034 112120B5 E79EC5F2 6A9F69DA 85D74EF6
A97A0B11 64EFA25F B1AE26BA 451CCDA7 A2E78433 9C447D56 2549A60B F0676294
BF580C91 9EC45702 5D3C7860 B98296C0 AB9FE5B1 D353882E 26C1F721 B41899D9
72B5A1D5 050B6845 36448010 AF8C7AFF 7CE8EACC B9B1FBBD C929D4F5 D499FB81
2924DF30 2CB3C450 23386297 9396B3A4 6CD0FF7F 1426711C 459297B6 5D1CEF66
C18751E0 94BF08F3 B2981C5C CE52D963 D5A4259A 64557E4D 1B9EFE0D 9A516D1E
6EC8BB37 066825AE A6361660 2BD7D116 25A06A90 739B4D0A 06EA872A 3AF9EBA1
2629BED6 7940561B D9374A89 D60F0D72 2C9FEB68 33EC53F0 B0FD76A2 047B66C9
0FCEB1D2 E22CC099 B9A4B93E
```

Table D-23: Upper Partial RSA Modulus 2

$$S'_b \| S'_c = X'\|Y'\|Z'\|M'_5\|M'_6\|M'_7\|M'_8\|M'_9\|M'_{10}\|M'_{11}\|M'_{12}$$

=
```
EE73E7D6 B3B34FBA A1393D02 A4742581 8DC84F86 736E9072 28BBE877 0203858D
8CF1837A FF5E6C22 13036AF3 D95C77E9 C2237D60 8CC4A9FB 97308BBF 9828612F
1599E261 5BCCDEDA 5930532F B3DD1172 78E49440 1433630E 7461C1DC 9B801B2E
552015A5 13FF7AE7 973EF44B 8352E4E0 4979B31E B600654D 51F4A381 CEBE3F0B
D099D130 D1456FAB E04A3E98 85C8C4FB 297B86B5 7752CD64 19809FE3 7E6286F0
7732D1E0 69A5B4E5 6670B8BB BAE5C211 742A131D 05711CF1 FE32AF93 3F1EEF22
4762E3AA DAC17C40 E448CA41 A879A03D 3CF665F2 39C7F3FE 82B384E8 35E7C9E8
BDEE30C2 68A21212 84789DF4 2F44906F 19B79026 464436E1 DA65FA0C 53A377FA
0D2B012B 7DDC2855 DAE5B551 51E28034 112120B5 E79EC5F2 6A9F69DA 85D74EF6
A97A0B11 64EFA25F B1AE26BA 451CCDA7 A2E78433 9C447D56 0549A60B F0676294
BF580C91 9EC45702 5D3C7860 B98296C0 AB9FE5B1 D353882E 26C1F721 B41899D9
72B5A1D5 050B6845 36448010 AF8C7AFF 7CE8EACC B9B1FBBD D129D4F5 D499FB81
2924DF30 2CB3C450 23386297 9396B3A4 6CD0FF7F 1426711C 459297B6 5D1CEF66
C18751E0 94BF08F3 B2981C5C CE52D963 D5A4259A 64557E4D 1B9EFE2D 9A516D1E
6EC8BB37 066825AE A6361660 2BD7D116 25A06A90 739B4D0A 06EA872A 3AF9EBA1
2629BED6 7940561B D9374A89 D60F0D72 2C9FEB68 33EC53F0 B0FD76AA 047B66C9
0FCEB1D2 E22CC099 B9A4B93E
```

Table D-24: Lower Partial RSA Modulus S_m

S_m =
```
0000000F 54A89517 6E4C295A 405FAF54 CEE82D04 3A45CE40 B155BE34 EBDE7847
85A25B7F 894D424F A127B157 A8A120F9 9FE53102 C81FA90E 0B9BDA1B A775DF75
D9152A80 257A1ED3 52DD49E5 7E068FF3 F02CABD4 AC97DBBC 3FA0205A 74302F65
C7F49A41 9E08FD54 BFAFC14D 78ABAAB3 0DDB3FC8 48E3DF02 C5A40EDA 248C9FF4
7482850C FDFBDD9B C55547B7 404F5803 C1BB8163 2173127E 1A93B24A FB6E7A80
450865DB 374676D5 76BA5296 CCC6C130 82D1AB36 521F1A8A D945466B 9EF06AF4
3A02D70B 7FB8B7DC 6D268C3D BA6898F6 552FA3FB B33DCBFA DA7B33FA 75D93AFE
262BD37A FF75995F D0E9774B A5A26A7C 443FF34E 461502A2 CB777E98 2D007375
14B88ED2 8D61F428 E88387DF 2BF02230 AD17A9D4 4FF36485 0A07DB42 A7826AC2
EE3899CA C3EC2747 21D476D9 6658F537 16676587 F8FF14DB 8DE6741A FA2206DB
A3B11828 BA87C6E1 E88A022F 1AA8DDD0 37EAB049 B5C7D305 3D0A63D7 861DEA07
B3D8B720 DE068CF4 7E657BB4 4450B85D 52F749D5 9572DF0C 0E3433B4 7C9AA19A
856F1DC3 CDADBAFB 143035C8 5A53AF57 22038F76 5C0D621B 66B69FFF FD091D4A
661A453B F1DAED1A 3A2341B3 7D7F623B 158F6EC0 2B49A253 64430FCB 5861483E
1E9543ED 2EE7E54A 4C108A6E 64194098 0EE60D14 AEE559AF 30037E75 B2309CE0
21FFE310 9BF20538 92AB0AE4 03516E2A B58067F7
```

Table D-25: RSA Moduli

$$
\begin{aligned}
n_1 &= S_b \| S_c \| S_m = p_1 \cdot q_1 \\
n_2 &= S'_b \| S'_c \| S_m = p_2 \cdot q_2
\end{aligned}
$$

Where

p_1 =
```
FF6E89C1 C29EC1B6 DCAC6227 EAD2226C E7E07D35 3F2296F7 940E6154 17A8363C
482171DE ECC75091 E5934F7E 7C1D6EAC 90B3A8D7 AD7C39CD A6364D79 CE8D9063
906933C9 64EAACF5 003B5D3A 1DF30C83 74C3CE80 4E54B4A8 DB6AEF33 166E282F
8425B5A9 9E640BC0 F87C3507 C888119E 2479DCF4 4E88538B CE9E7BC3 A7D7A454
78F69937 9FA845DB 43636513 FB3C2468 D32AB56F FD4A49C4 D73EB135 6C6FFEAA
921B8A27 6DF4CA34 512835C4 CCC3E6B2 77A689F5 73009A2B 90E985FD E63CE7F3
59D30AC1 92A2C97F 05C9DCEC 46B17355 0F926164 9F4613E8 B349B5C4 CB090692
8278DBFF 534B02E8 5A305B93 069BA793 5893BE68 F9C197
```

p_2 =
```
F134344B 72A468C3 EA7A5B2F 97CDFE2F DB9194CE 47B03C85 9A4E8A0F BE2B1B1B
55CE1E96 5409BB5F 0F07F2CF B67C3FE3 27853D37 8D0038A6 94A16AAD 84038E18
D69746A4 C1126D21 D5839065 F0885C60 BB174114 B76B003F 368AB2EF 6FF46A59
34DBCBE1 1517FD9E 6F418A06 F4F3BE6A ABB77B2F 999B4FE9 76C8096E C0133761
AFD0149B 4816EAC9 2C06E1AF 60C05F19 FDA2A23A B4A5CA4A 05403033 EB65FB3C
648B0536 09C5C43A 4EE308CA BA8E639C EB7C297D 56A398DD C35E42B7 31AFC9C0
22414B8F 6A94A280 E4D9EF28 F995553B 3FA3E308 19911F98 43276163 91336C18
85EC8062 A1D2CA68 990C0174 561DAE3F 6B3C7378 2D53BD
```