



桂林电子科技大学
GUILIN UNIVERSITY OF ELECTRONIC TECHNOLOGY

应用密码学课程报告

课题名称: 任意文件 MD5 碰撞

课程名称: 应用密码学

学 院: 计算机与信息安全学院

组 长: 1900300223 卢 畅

成 员: 1900300222 刘 杨

1900800609 黄亮杰

1900300104 秦洪微

1900301217 李新宇

指导老师: 张瑞霞

2021 年 11 月 26 日

团队任务分工表

题目	任意文件 MD5 码碰撞		
组长	林楷浩	指导教师	张瑞霞
序号	学号	姓名	个人任务描述
1	1900300223	卢畅	主要负责项目总体架构，团队任务分配。 fastcoll 代码复现以及后期文档整合与校验。
2	1900300222	刘杨	主要负责 MD5 基础知识部分文档与结构图； 代码注释编写。
3	1900800609	黄亮杰	主要负责资料查询，应用可视化界面编写以及 部分文档编写。
4	1900300104	秦洪微	主要负责 MD5 碰撞攻击部分文档与结构图； 代码注释编写。
5	1900301217	李新宇	主要负责 MD5 碰撞攻击部分文档与结构图； 代码注释编写。

任意文件MD5碰撞

摘要

Hash函数是信息安全领域中一个非常重要的基本工具,它在数字签名、身份认证等领域中有着广泛的应用。MD5算法作为Hash函数大家庭中的一员,是MD结构的典型代表,广泛应用于信息安全领域。因此,通过对MD5算法的解析可以掌握Hash函数的基本分析方法,对其他Hash函数的分析有着重要的参考意义。

本项目对MD5算法的攻击技术进行了分析和研究。首先系统总结了对该算法进行碰撞攻击的整体思想,运用了比特跟踪技术对差分路径进行分析和控制,利用消息修改技术提高搜索到产生碰撞的明文对的概率。本项目在基于fastcoll的基础上对代码进行优化和修改并给出了MD5碰撞的实际运用场景。

关键字：MD5 碰撞攻击

目录

第一章 绪论	1
1.1 项目背景	1
1.2 项目简介	1
第二章 MD5 基础知识	2
2.1 MD5 内容介绍	2
2.2 MD5 算法	2
2.2.1 整体结构描述	2
2.2.2 压缩函数	3
2.3 MD5 的差分分析	6
2.3.1 三种差分的概念引入	6
2.3.2 循环左移的结果分析	7
第三章 MD5 的碰撞攻击	8
3.1 碰撞攻击的整体思想	8
3.1.1 引入明文差分	8
3.1.2 选择差分路径	8
3.1.3 确定充分条件	9
3.2 比特跟踪技术	10
3.2.1 符号说明	10
3.2.2 比特跟踪技术的应用	11
3.3 消息修改技术	14
3.3.1 单消息修改技术	14
3.3.2 多消息修改技术	15
3.3.3 程序实现	16
3.3.4 消息修改技术总结	18
3.3.5 碰撞设计的基本原则	18
3.3.6 碰撞攻击的评价	19
第四章 项目功能介绍	20
4.1 环境配置	20
4.1.1 代码下载与编译	20
4.1.2 boost 环境配置	20
4.2 功能概述	20
4.3 任意文件 MD5 碰撞	20

4.4 项目在 CTF 中的运用.....	23
4.4.1 强网杯 CTF 比赛.....	23
4.4.2 强网杯中的 MD5 碰撞	23
4.4.3 构造	23
4.4.4 测试	23
第五章 总结.....	25
参考文献.....	26

第一章 绪论

1.1 项目背景

Hash函数在数字签名系统中扮演着重要的角色。“Hash函数”一词最初来源于计算机科学,表示可以将任意长度的字符串压缩成固定长度的字符串的函数,Hash函数的输出结果,称为Hash值,又称为消息摘要、指纹等。

通俗地讲,Hash函数用于压缩消息,是将任意长的消息映射为固定长度的Hash值。消息的Hash值可以看作是数字指纹:像实际生活中指纹可以验证一个人身份的唯一性一样,Hash函数可用于验证一个消息的唯一性。也就是说,给定某一消息和Hash值,我们可以判断此消息与Hash值是否匹配。此外,类似于我们在嫌疑犯数据库里比较一系列犯罪现场出现的指纹一样,我们可以从有限的消息中检测某些Hash值与哪些原始消息相匹配。然而,对于给定的Hash值不可能恢复出其原始消息。Hash函数有三个安全特性,包括抗碰撞攻击、抗原象攻击和抗第二原象攻击。由于这些特性,Hash函数在公钥密码、数字签名、完整性检验、身份认证等领域中有着广泛的应用。

目前,Hash函数主要有MDx系列和SHA系列,MDx系列包括MD4[1]、MD5[2]、HAVAL[3]、RIPEMD[4]等;SHA系列包括SHA-0[5]的、SHA-1[6]、SHA-256、SHA-384[7]等,这些Hash算法体现了目前主要的Hash函数设计技术。

一些针对Hash函数的更为有效的攻击方法出现在2004年。Eli Biham和Rai提出了对SHA-0算法的几乎完全碰撞。AJoux给出了一个由4个明文块构成的SHA-0的碰撞。在这些成果中,最为显著的是在2004年国际密码学大会上,王小云等人宣布的对于一系列Hash函数的碰撞结果,包括MD4[7]、MD5[8]、HAVAL-128[9]和RIPEMD算法,其中可以找到MD4[7]和RIPEMD算法碰撞的复杂性分别低于 2^8 和 2^{18} 。王小云提出了一套新的针对MDx系列的Hash函数的分析技术,同时给出了得到满足差分路线的充分条件的方法,以及如何使用明文修改技术来提高碰撞攻击的成功概率。2005年,王小云等应用此技术对MD5[8]、SHA-0[5]、SHA-1[6]算法进行碰撞攻击,取得了很好的效果,在普通的个人电脑上就可以找到相关碰撞的实例。这一攻击技术对现有的Hash函数提出了严峻的挑战,促进了新的Hash算法的开发研究。

1.2 项目简介

本项目研究的主要内容是针对MD5算法的结构特点,对其进行碰撞攻击,分析总结其攻击的思路和方法。项目对文献[14]基于选择前缀碰撞攻击方法进行复现使用实现了任意文件的碰撞,并对多种格式的文件进行碰撞。测试碰撞生成文件的MD5数值以及文件的正常工作状态。

消息处理过程如下:

1、对消息的填充。对消息的填充,使得其比特长在模512下为448,即填充后消息的长度为512的某一倍数减去64,留出64比特以备第2步使用。步骤1是必须的,即使消息长度已经满足要求,仍需要填充。例如,消息长为448比特,则需要填充512比特,使其长度变为960,因此填充的比特数大于等于1,小于等于512。填充的方法是确定的:第1位为1,后面的位都为0。

2、附加消息的长度。用步骤1留出的64比特以little-endian方式来表示消息被填充前的长度。如果消息的长度大于 2^{64} ,则以 2^{64} 为模数取模。Little-endian方式是指数据的最低有效字节(byte)(或者最低有效位)的优先顺序存储数据,即将最低有效字节(或者最低有效位)存于第地址字节(或者位)。相反的存储方式称之为big-endian方式。

前两步执行完后,消息长度为512的倍数(设为 L 倍),因此可将消息表示为分组长为512比特的一系列分组 $Y_0Y_1\cdots Y_t$,而每一分组有可表示为16个32比特长的字,这样消息中的总字数为 $N = L \times 16$,因此消息又可按字表示为 $M[0, 1, \cdots N - 1]$ 。

3、对MD缓冲区初始化。算法使用128比特长的缓冲区以存储中间结果和最终Hash值,缓冲区可以表示为4个32比特长的寄存器(A, B, C, D),每个寄存器都以little-endian方式存储数据,其初值(以存储方式)为 $A = 01234567, B = 89ABCDEF, D = 10325476$ 。

4、以分组为单位对消息进行处理。每一分组 $Y_q(q = 0, \cdots, L - 1)$ 都经一压缩函数处理,压缩函数是整个MD5算法的核心,共有4个不同的布尔函数,分别为F函数、G函数、H函数、I函数。每一轮的输入为当前处理的消息分组 Y_q 和缓冲区当前值,输出仍放在缓冲区中以产生新的 A, B, C, D 。每一轮处理还需要加上常数表中的常数。常数表中一共有64个元素,第 i 个常数值为 $2^{32} \times \text{abs}(\sin(i))$ 的整数部分。第四轮的输出再与第一轮输入值相加,相加的结果即为压缩函数的输出值。

5、输出。消息 L 个分组都被处理完后,最后一个压缩函数输出值即为整个MD5运算的最终Hash值,即消息摘要。

2.2.2 压缩函数

压缩函数是MD5算法的核心部件,对压缩函数的分析是对算法进行成功攻击的重要前提。该压缩函数中有4轮处理过程,每一轮有对缓冲区ABCD进行16步迭代运算,每一步的运算形式(如图2.2所示)为:

$$a = b + ((g + \text{func}(b, c, d) + X[i] + T[i]) \lll k) \quad (3)$$

其中 a 、 b 、 c 、 d 为缓冲区的 4 个字, $func$ 代表具体的布尔函数, 为 F 、 G 、 H 、 I 之一; $X[i]$ 为该步使用的明文字; $T[i]$ 为该步使用的常数; K 为该步循环左移的比特数。

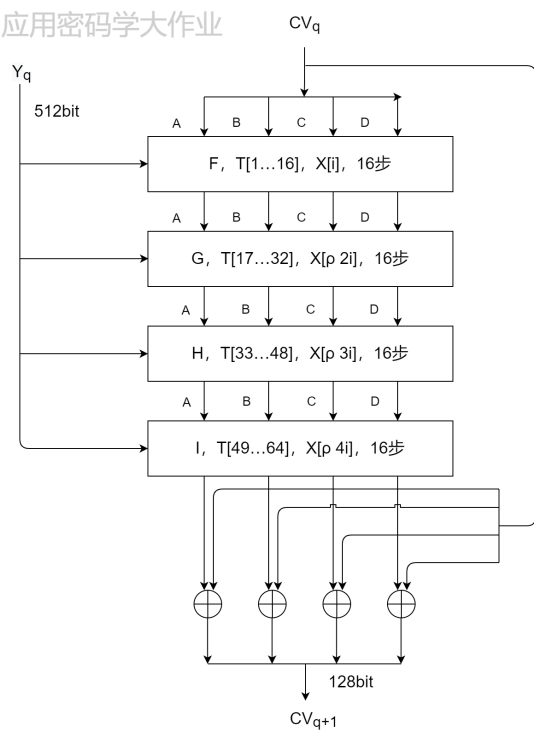


图2.2 迭代结构

在计算新的 a 的表达式中, “+” 为模 2^{32} 的加法。明文字(消息字)在 4 轮处理的过程中, 每一轮以不同的次序使用 16 个明文字, 第一轮按照原来顺序进行使用。第二轮到第四轮分别对这 16 个字的次序进行变换(如表 2.1 所示), 然后以新的次序使用这 16 个字。

表2.1 明文字转换次序

应用密码学大作业

轮数	明文字的次序
1	i
2	$(1+5*i) \bmod 16$
3	$(5+3*i) \bmod 16$
4	$(7*i) \bmod 16$

4 轮处理过程分别使用不同的布尔函数 F 、 G 、 H 、 I , 每个布尔函数的输入为 3 个 32 比特的字, 输出为一个 32 比特的字, 其中的运算为逐比特的逻辑运算, 即输出的第 n 个比特是 3 个输入的第 n 个比特的函数。四个布尔函数, 如表 2.2 所示:

表2.2 四个布尔函数

应用密码学大作业

轮数	布尔函数	func (b,c,d)
1	F(b,c,d)	$(b \wedge c) \vee (\neg b \wedge d)$
2	G(b,c,d)	$(b \wedge d) \vee (c \wedge \neg d)$
3	H(b,c,d)	$b \oplus c \oplus d$
4	I(b,c,d)	$c \oplus (b \vee \neg d)$

四个布尔函数的真值如表 2.3 :

应用密码学大作业

b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

表2.3 布尔函数真值表

2.1.3移位和明文字顺序表

在对MD5算法进行碰撞攻击的过程中,必须考虑移位对整个差分路径的影响。在碰撞攻击过程中,明文字的编排顺序是差分引入的重要考虑因素之一;在原像攻击过程中,明文字的编排顺序很大程度上确定了初始结构和过渡结构的位置。

表2.4 移位比特表

应用密码学大作业

步骤	位移比特数
1, 2,, 15	7 12 17 22 7 12 17 22 7 12 17 22 7 12 17 22
16, 17,, 31	5 9 14 20 5 9 14 20 5 9 14 20 5 9 14 20
32, 33,, 47	4 11 16 23 4 11 16 23 4 11 16 23 4 11 16 23
48, 49,, 63	6 10 15 21 6 10 15 21 6 10 15 21 6 10 15 21

表2.5 明文字顺序

应用密码学大作业

步骤	明文字顺序
1, 2,, 15	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
16, 17,, 31	1 6 11 0 5 10 15 4 9 14 3 8 13 2 7 12
32, 33,, 47	5 8 11 14 1 4 7 10 13 0 3 6 9 12 15 2
48, 49,, 63	0 7 14 5 12 3 10 1 8 15 6 13 4 11 2 9

2.3 MD5的差分分析

2.3.1 三种差分的概念引入

差分分析是破译Hash函数最有效的工具之一,其中涉及到异或差分、符号差分 and 模差分等几个重要的概念。

异或差分:设 X 和 X' 是两个32位的数,其异或差分为两个数的比特位的异或运算,表示为 $X \odot X'$ 。异或差分能反映两个数的对应位的相同与否。

符号差分:同样,设 X 和 X' 是两个32位的数,为了更好的体现这两个数的某一位的差别,在异或差分的基础上进一步引入符号差分“+”、“-”号。例如, $X' - X = [5, 6, 7, 8, \dots, -27]$,这说明 X 和 X' 从第5到第27位都不同,而且从符号可以看出 X 的第5到第26位是0,第27位是1;而 X' 的第5到第26位是1,第27位是0。符号差分可以很容易看出两个数的各位的差别。

模差分:同样,设 X 和 X' 是两个32位的数,模差分是将两个数的差别用整数减法体现出来。例如, $X' - X$ 用符号差分表示为 $[5, 6, 7, 8, \dots, -27]$,此时用模差分表示为 $X' - X = -2^4$ 。

符号差分的扩展在MD5算法破译过程中具有广泛的应用,利用符号差分的扩展,可以达到自己所需要的差分目标。例如,对 2^k 、 -2^k 可以进行如下扩展:
 $2^k = 2^{k+1} - 2^k = 2^{k+2} - 2^{k+1} - 2^k = \dots = 2^{n-1} - 2^{n-2} - 2^{n-3} - \dots - 2^k$, 其中
 $0 \leq k < n \leq 32$

$-2^k = -2^{k+1} + 2^k = -2^{k+2} + 2^{k+1} + 2^k = \dots = -2^{n-1} + 2^{n-2} + 2^{n-3} - \dots + 2^k$, 其中
 $0 \leq k < n \leq 32$

在MD5的运算中,第32位的位置既特殊又重要,其“+”和“-”在某种程度上是可以看成一致的。鉴于这一位的特殊性,可以将上面两个等式进行进一步扩展:

$$\begin{aligned}
 2^k &= 2^{k+1} - 2^k = 2^{k+2} - 2^{k+1} - 2^k \\
 &= \dots = 2^{31} - 2^{30} - 2^{29} \dots - 2^k \\
 &= -(2^{31} + 2^{30} + 2^{29} + \dots + 2^k) \\
 -2^k &= -2^{k+1} + 2^k = -2^{k+2} + 2^{k+1} + 2^k \\
 &= \dots = -2^{31} + 2^{30} + 2^{29} - \dots + 2^k \\
 &= (2^{31} + 2^{30} + 2^{29} - \dots + 2^k)
 \end{aligned} \tag{4}$$

上面的差分扩展还可以将两个元素联合起来考虑:

$$\begin{aligned}
&= [-(k+2), (k+1), -k] = [-(k+1), -k] \\
&= [-(k+3), (k+2), (k+1), -k] \\
&= [-(k+3), (k+2), k] = \dots
\end{aligned} \tag{5}$$

符号差分扩展在MD5差分路径的控制上有着至关重要的作用,一般情况下,在模差分确定后,可以根据具体需要确定符号差分。

2.3.2 循环左移的结果分析

MD5算法在运算过程中有大量的左循环移位,所以在引入差分之后,符号差分的左循环移位会出现几种不同的情况,需要分别进行分析处理。

例如,文献[12],设 $2^k = [k+x, -(k+x-1), \dots, -k]$,其中 $1 \leq x \leq 32-k$,设循环左移的位数为 s (用 $\lll s$ 表示),分三种情况讨论移位后的结果:

当 $k+x+s \leq 32$,移位的结果为 $(2^k \lll s) = 2^{k+s}$;

当 $k+x+s > 32$ 且 $k+x \leq 32$,移位的结果为 $(2^k \lll s) = 2^{k+s} + 1$;

当 $k+x > 32$,移位的结果为 $(2^k \lll s) = 2^{k+s-32}$ 。

证明过程如下:

1. 当 $k+x+s \leq 32$ 时,

$$\begin{aligned}
(2^k \lll s) &= [k+x, \\
&\quad -(k+x-1), -(k+x-2, \dots), -k,] \lll s \\
&= [k+x+s, -(k+x+s-1), \dots, -(k+s)] \\
&= 2^{k+s} \bmod (2^{32})
\end{aligned} \tag{6}$$

2. 当 $k+x+s > 32$ 且 $k+x \leq 32$ 时,

$$\begin{aligned}
(2^k \lll s) &= [k+x, \\
&\quad -(k+x-1), -(k+x-2, \dots), -k,] \lll s \\
&= [-32, \dots, -(k+x), k+x+s-32, \\
&\quad -(k+x+s-32-1), \dots, -1] \\
&= [32, \dots, -(k+x), k+x+s-32, \\
&\quad -(k+x+s-32-1), \dots, -1]
\end{aligned} \tag{7}$$

3. 当 $k+x > 32$ 时,

$$\begin{aligned}
(2^k \lll s) &= [k+x, \\
&\quad -(k+x-1), -(k+x-2, \dots), -k,] \lll s \\
&= [k+x+s-32, -(k+x+s-32-1), \\
&\quad \dots, -(k+s-32)] \\
&= (2^{k+s-32}) \bmod (2^{32}) \\
&= (2^{k+s} + 1) \bmod (2^{32})
\end{aligned} \tag{8}$$

4. 当 $k + s > 32$ 时,

$$\begin{aligned}
 (2^k \lll s) &= [k + x, \\
 &\quad - (k + x - 1), -(k + x - 2, \dots), -k,] \lll s \\
 &= [k + x + s - 32, -(k + x + s - 32 - 1), \\
 &\quad \dots, -(k + s - 32)] \\
 &= (2^{k+s-32}) \bmod (2^{32})
 \end{aligned} \tag{9}$$

同理,设 $-2^k = [-(k + x), (k + x - 1), \dots, k]$,其中 $1 \leq x \leq 32 - k$,同样分三种情况讨论:当 $k + x + s \leq 32$,移位的结果为 $(-2^k \lll s) = -2^{k+s}$;
当 $k + x + s > 32, k + s \leq 32$,移位的结果为 $(-2^k \lll s) = -(2^{k+s} + 1)$;当 $k + s > 32$,移位的结果为 $(-2^k \lll s) = -2^{k+s-32}$ 。

第三章 MD5的碰撞攻击

3.1 碰撞攻击的整体思想

破译MD5算法是个系统工程。在破译过程中,首先要从整体上确立一个框架,确定一个产生碰撞的总体思路;其次,根据整体思路应用明文修改等相关技术和非线性函数的特性等来实现差分路径;最后,尽可能简化实现差分路径的充分条件,降低计算复杂度,提高计算机“搜索”到碰撞的概率。

3.1.1 引入明文差分

破译 MD5 算法的关键之一是在明文中引入差分。好的明文差分有以下几个特点: 首先,要有尽可能多的“自由字”,这样可以放宽产生碰撞的充分条件,提高碰撞概率;其次,明文中第一次出现差分的位置应该尽量远离第一个字,这对于消息修改技术至关重要,可以使明文存在更大的修改空间,同时也有利于计算机的成功搜索;再次,实现碰撞所需要的充分条件越少越好,这样可以有效的降低计算复杂度;最后,明文的“块数”越少,说明实现碰撞的技术水平越高。目前,发表的三种MD5算法的碰撞实例,都是采用“两个明文块”,即1024位的明文,其中文献[10]的两个明文块的差分的位置都是位置对称、符号相反,这种差分的引入是针对Hash函数MD结构的特点而设计的,它能使得第一明文块和第二明文块分别产生的差分值能够位置对称,符号相反(第32位除外),最终的Hash值是将两个值相加。这样,两个明文块产生的差分最终将抵消(第32位以溢出的形式抵消),从而产生了碰撞。文献[12]差分的引入摆脱了对称的模式,但其差分的消除仍遵循上述原则。最后要说明的是,以上三种MD5算法的碰撞实例,在寻找碰撞的过程中,都需要利用明文修改技术来确保得到碰撞所需的明文。

3.1.2 选择差分路径

破译MD5算法的关键之二是要选择好差分路径。差分路径的控制很大程度上是利用符号差分的扩展来实现的,使得整个差分按照整体的需要进行。选择差分路径时,要尽量减少差分中的汉明重量。一般情况下,尽量把差分引到第32位,由于该位置的特殊性,容易消除差分。

在选择差分路径的过程中,必须关注每一步运算中差分的继承和非线性函数的位运算。以 MD5 第 12 步为例, $b_3 = c_3 + ((b_2 + F(c_3, d_3, a_3) + m_{11} + t_{11}) \lll 22)$, 令 $\Sigma_{11} = b_2 + F(c_3, d_3, a_3) + m_{11} + t_{11}$, 其中 t_{11} 是常数。 b_3 的差分有以下两个来源:

- 1、直接继承 c_3 的差分。
- 2、间接继承 Σ_{11} 的差分:第一是 b_2 的差分,第二是 $F(c_3, d_3, a_3)$ 的差分,第三是 m_{11} 的差分(如果 m_{11} 有差分的前提下)。

在计算 b_3 运算中,只有 F 函数是位运算,其他都是在模 2^{32} 下的加法运算。在这个过程中,首先充分关注 F 函数,利用 F 函数的性质,产生下面步骤所需要的差分,同时消除不需要的差分;其次符号差分的扩展很大程度上是依靠 Σ_{11} 来实现的。在模差分确定的前提下,符号差分有多种可能,必须根据需要进行选择;最后结合第32位的特殊性和符号差分的扩展,跟踪循环左移对这个差分继承的影响。

3. 1. 3 确定充分条件

破译MD5算法的关键之三是控制差分路径的充分条件。充分条件越少,对明文的限制程度越小,越容易找到碰撞。在MD5中利用 F 函数、 G 函数、 H 函数和 I 函数的性质,循环左移的性质和差分路径的整体需要,来确定差分的充分条件。

在确定充分条件的过程中,必须利用非线性函数的性质。MD5算法中有 F 函数、 G 函数、 H 函数和 I 函数这四个非线性函数,现在以 F 函数和 H 函数为例,总结非线性函数的特性^[31]。

从 $F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$ 的表达式,可以看出 F 函数是一个选择函数。选择函数的意思是: F 函数的值的选取是根据 X 的值来决定选取 Y 或者 Z 的值:即如果 $X = 1$, F 函数的值等于 Y 的值;如果 $X = 0$, F 函数的值等于 Z 的值。 X 的值对 F 函数的值起到一个选取的作用。根据上述所说的 F 函数的选取的性质,我们可以总结出 F 函数的三个性质:

- 1、 $F(X, Y, Z) = F(\neg X, Y, Z)$, 当且仅当 $Y = Z$;
- 2、 $F(X, Y, Z) = F(X, \neg Y, Z)$, 当且仅当 $X = 0$;
- 3、 $F(X, Y, Z) = F(X, Y, \neg Z)$, 当且仅当 $X = 1$;

从 $H(X, Y, Z) = X \odot Y \odot Z$ 的表达式,可以看出 H 函数是个对称函数,该函数具有以下两个性质:

- 1、X, Y, Z 这三个参数中相同的位有奇数个1, H 函数的值对应的位为1;
 - 2、X, Y, Z 这三个参数中相同的位有偶数个1, H 函数的值对应的位为0。
- 总之,在寻找MD5碰撞的过程中,上述三个方面都很关键,他们相辅相成,相互协调,必须统一起来整体考虑。

3.2 比特跟踪技术

在MD5碰撞攻击的过程中,在差分引入之后,必须应用比特跟踪技术对差分进行比特级别的跟踪,以确保有效地控制差分路径,最终实现碰撞。
就MD5的第一轮而言,以四个连续步将为例:

$$\begin{aligned}
 a &= b + ((a + F(b, c, d) + m_i + t_i) \lll s_i) \\
 d &= a + ((d + F(a, b, c) + m_{i+1} + t_{i+1}) \lll s_{i+1}) \\
 c &= d + ((c + F(d, a, b) + m_{i+2} + t_{i+2}) \lll s_{i+2}) \\
 b &= c + ((b + F(c, d, a) + m_{i+3} + t_{i+3}) \lll s_{i+3})
 \end{aligned} \tag{10}$$

在这每一步中,只有 F 函数的运算是位运算,除此以外的运算都是模 2^{32} 的加法运算。因此,在 F 函数中,其三个参数某一比特的变化,最终只会影响 F 函数的相应的比特位; F 函数以外的运算,由于可能存在进位的问题,所以影响的可能不只是变化的那一比特,而且可能是影响其周围的一连串的比特位,影响的范围可能很大,这就需要依靠比特跟踪技术来监控。在整个运算过程中,第32比特位置特殊,其进位将会产生溢出,这个性质在比特跟踪过程中至关重要。下面将以文献[12]为例,介绍比特跟踪技术在碰撞攻击中的应用。

3.2.1 符号说明

- 1、 $M = (m_0, m_1, \dots, m_{15})$ 和 $M' = (m'_0, m'_1, \dots, m'_{15})$ 代表相对应的两个512比特的明文块, $\Delta M = (\Delta m_0, \Delta m_1, \dots, \Delta m_{15})$ 代表两个相对应的明文块的差分,其中每个 m_i 为32比特的字。
- 2、 a_i, d_i, c_i, b_i 分别代表第 $(4i - 3)$ 步,第 $(4i - 2)$ 步,第 $(4i - 1)$ 步,第 $4i$ 步的输出,其中 $1 \leq i \leq 16$;同样的方法定义 a'_i, b'_i, c'_i, d'_i 。
- 3、 $a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j}$ 分别代表 a_i, b_i, c_i, d_i 的第 j 比特,其中最高位为第32比特,最低位为第1比特。
- 4、 $\Phi_{i,j}, \Psi_{i,j}$, 分别表示 Φ_i 和 Ψ_i 的第 j 比特,其中 Φ_i 表示第 i 步的逻辑函数, Ψ_i 表示循环左移的整体的表达式。例如,第8步中, $\Phi_7 = F(c_2, d_2, a_2) = (c_2 \wedge d_2) \vee (\neg c_2 \wedge a_2)$, $\Psi_7 = b_1 + F(c_2, d_2, a_2) + m_7 + t_7$
- 5、 $\Delta x_{i,j} = x'_{i,j} - x_{i,j} = \pm 1$, 这是表示第 j 比特的差分。如果值为1,说明 x'_i 的第 j 比特是1而 x_i 的第 j 比特是0;如果值是0,说明 x'_i 的第 j 比特是0而 x_i 的第 j 比特是1,其中 x 可以是 a, b, c, d, Φ, Ψ 。

6、 $\Delta x_i[j1, j2, \dots, jl] = x_i[j1, j2, \dots, jl] - x_i$, 表示 x_i 的第 $j1, j2, \dots, jl$ 比特的变化。 $x_i[j1, j2, \dots, jl]$ 表示具体第 $j1, j2 \dots jl$ 的比特的变化, 其中“+”表示该比特从0变成了1;“-”表示该比特从1变成了0。

3. 2. 2 比特跟踪技术的应用

文献[2]是用两个明文块 (M_0, M_1) , 明文长度共为1024比特, 对应的差分明文块为 (M'_0, M'_1) , 经过两次MD5运算, (M_0, M_1) 和 (M'_0, M'_1) 产生相同的Hash值, 这就成功找到了碰撞。本文通过对第一次MD5迭代运算的前8步进行研究, 所以只关注 M_0 和 M'_0 , 其中:
 $\Delta M_0 = M'_0 - M_0 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, 2^{15}, 0, 0, 2^{31}, 0)$ 。从这里可以看出, m_4 开始引入差分, 本文结合文献[12], 以前8步为例, 介绍比特跟踪技术对差分路径的控制。

第5步分析:

第5步的表达式: $a_2 = b_1 + ((a_1 + F(b_1, c_1, d_1) + m_4 + t_4) \lll 7)$, 开始引入了差分, $\Delta m_4 = 2^{31}$, 在前4步的过程中, 明文没有引入差分, 所以 a_1, b_1, c_1, d_1 都没有出现差分, 即
 $a_1 = a'_1, d_1 = d'_1, c_1 = c'_1, b_1 = b'_1$ 。 $\Delta a_2 = (\Delta \Psi_4[31] \lll 7) = ((-2^{31}) \lll 7) = -2^6$ 。
 根据基础知识, 第32比特的“+”和“-”是相同的, 所以此时明文引入的差分 2^{31} , 看成 -2^{31} 来处理, 经过循环左移, Δa_2 的模差分为 -2^6 , 为了继续进行下面的步骤, 需要将这个模差分进行扩展, 满足 $a_2[+7, +8, +9, \dots, +22, -23]$ 。此时要求
 $a_{2,7} = a_{2,8} = a_{2,9} = \dots = a_{2,22} = 0, \quad a_{2,23} = 1$ 。

第6步分析:

第6步的表达式: $d_2 = a_2 + ((d_1 + F(a_2, b_1, c_1) + m_5 + t_5) \lll 12)$, 这步明文
 $\Delta m_5 = 0, \Delta d_2 = \Delta a_2 + (\Delta \Psi_5 \lll 12)$, Δd_2 的差分由两部分组成: 首先, Δd_2 继承了 $\Delta a_2 = -2^6$ 这部分的值; 其次, $(\Delta \Psi_5 \lll 12)$ 产生了后面所需要的差分
 $2^{23} + 2^{31}$ 。

根据前面的基础知识, 为了达到这个目标, 需要充分利用F函数的性质:

1、利用 $F(X, Y, Z) = F(\neg X, Y, Z)$, 当且仅当 $Y = Z$ 。第6步中 $b_{1,i} = c_{1,i}$, 这个条件确保 a_2 的第 i 比特的变化不会影响 d_2 , 其中
 $i = 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23$ 。

2、 $2^{23} + 2^{31}$ 是由 $\Delta \Psi_5[12]$ 和 $\Delta \Psi_5[20]$ 经过循环左移12比特形成的。此时由于没有差分的扩展, 满足
 $\Delta \Psi_5[12] = \Delta \Phi_5[12] = 1, \Delta \Psi_5[20] = \Delta \Phi_5[20] = 1$ 。此时要求
 $b_1[12] - c_1[12] = 1, b_1[20] - c_1[20] = 1$, 即
 $b_1[12] = 1, c_1[12] = 0, b_1[20] = 1, c_1[20] = 0$ 。

第7步分析:

第7步表达式: $c_2 = d_2 + ((c_1 + F(d_2, a_2, b_1) + m_6 + t_6) \lll 17)$, 这步明文
 $\Delta m_6 = 0, \Delta c_2 = \Delta d_2 + (\Psi_6 \lll 12)$, 同第6步的分析一样, Δc_2 的差分由两部分组成:

1、 Δc_2 继承了 $\Delta d_2 = -2^6 + 2^{23}$ 这部分的值,而且都进行了扩展。

2、研究 $F(d_2, a_2, b_1)$ 部分,根据 F 函数的性质 $\Delta d_2, \Delta a_2$, 可以得出:
 $\Delta \Phi_6 = [-23, 22, 21, 20, 19, 18, 17, 16, 14, 13, 12, 11]$, 在经过
 $\Psi_6 = c_1 + \Phi_6 + m_6 + t_6$, 异或差分进行了变化,
 $\Delta \Psi_6 = -2^{16} + 2^{14} + 2^{13} + 2^{12} + 2^{11}$, 即 $\Delta \Psi_6 = [-16, 14, 13, 12, 11]$ 。

3、经过循环左移 17 比特, $\Delta \Psi_6 \lll 17 = [-1, 31, 30, 29, 28]$ 。 Δc_2 继承了
 $\Delta \Psi_6 \lll 17$ 的比特变化,其中,第 1 比特的差分进行了扩展了 6 个比特,使得
 $c_2[-6, 5, 4, 3, 2, 1]$,其他比特没有进行扩展。

4、 Δc_2 还继承了 $\Delta d_2 = 2^{31}$ 这部分的值,联合 $\Delta \Psi_6 \lll 17 = [31, 30, 29, 28]$,这些
部分共同构建了 $c_2[32, 31, 30, 29, 28]$ 。

详细解析第 8 步中各比特的由来:

第 8 步在文献 [1] 中是个关键的步骤,控制好该步的差分对最终产生碰撞有着重
要的作用。该步的差分是由前 7 步差分共同作用产生的,即
 $(\Delta c_2, \Delta d_2, \Delta a_2, \Delta b_1) \rightarrow \Delta b_2$ 。为了控制好差分路径,使得经过第 8 步的运算以
后,达到 $b_2' = b_2[1, 16, -17, 18, 19, 20, -21, -24]$ 的差分特性,必须控制好前面
的 $\Delta c_2, \Delta d_2, \Delta a_2, \Delta b_1$ 的值。前 7 步的运算为第 8 步提供了以下前提条件:

$$\begin{aligned} b_1' &= b_1; \\ a_2' &= a_2[7, \dots, 22, -23]; \\ d_2' &= d_2[-7, 24, 32]; \\ c_2' &= c_2[1, 2, 3, 4, 5, -6, 7, 8, 9, 10, 11, -12, \\ &\quad -24, -25, -26, 27, 28, 29, 30, 31, 32] \end{aligned} \quad (11)$$

第 8 步的运算如下:

$$\begin{aligned} b_2 &= c_2^{NF} + ((b_1 + F(c_2^F, d_2, a_2) + m_7 + t_7) \lll 22) \\ b_2' &= c_2^{NF'} + ((b_1' + F(c_2^{F_2}, d_2', a_2') + m_7' + t_7') \lll 22) \end{aligned} \quad (12)$$

b_2 的表达式中有两个 c_2 ,为了更好的区分,用 c_2^{NF} 和 c_2^F 分别表示。从上面两个
表达式可以看出:在第 8 步运算中明文没有引入差分,所以 $m_7' = m_7$;常数 t_7 是个
定值,所以 $t_{77} = t_7$;第 4 步的运算中没有引入差分,所以 $b_1' = b_1$ 。设
 $\Phi_7 = F(c_2, d_2, a_2) = (c_2 \wedge d_2) \vee (\neg c_2 \wedge a_2)$, 同样的定义
 Φ' ; $\Psi_7 = b_1 + F(c_2, d_2, a_2) + m_7 + t_7$,同样的定义 Ψ' 。

应用 F 函数的性质,对第八步的 F 函数进行研究:

1、采用 F 函数的第一个性质: $F(X, Y, Z) = F(\neg X, Y, Z)$,当且仅当 $Y = Z$ 。第 8
步中 $d_{2,i} = a_{2,i}$,这个条件确保 c_2^F 的第 i 比特的变化不会影响 b_2 , 其中 $i =$
 $1, 2, 4, 5, 25, 27, 29, 30, 31$ 。

2、采用 F 函数的第三个性质: $F(X, Y, Z) = F(X, Y, \neg Z)$, 当且仅当 $X = 1$ 。第 8 步中 $c_{2,i} = 1$, 这个条件确保了 a_2 的第 i 比特的变化不会影响 b_2 , 其中 $i = 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23$ 。

对第 8 步输出值 b_2 不变化的部分进行分析:

1、 $c_2[17] = 0, a_2[17] = 1$ 确保 b_2 的第 7 比特没有变化。 c_2^{NF} 的第 7 到第 12 比特的变化, 按照前面的基础知识, 可以将其统一起来考虑, 尽管它的异或差分有 6 比特, 但是它的模差分就只有 -2^6 。
 $\Delta b_2[7] = c_2^{NF}[7, \dots, 11, -12] + (\Delta \Psi_7[17] \lll 22) = -2^6 + (\Delta \Psi_7[17] \lll 22)$ 。
 为了保持 b_2 的第 7 比特不变化, 现在考虑 $\Delta \Psi_7[17]$ 。虽然 $\Delta b_1 = 0, \Delta t_7 = 0, \Delta m_7 = 0$, 但是由于 Ψ_7 表达式是在模 2^{32} 下的运算, 所以其中的进位可能会扩散影响到其他比特。这需要运用明文的修改技术, 这将在后面的内容进行讨论。根据文献 [12] 表二提供的数据程序进行测试, 在第 8 步中, $\Delta \Psi_7[17] = \Delta \Phi_7[17]$, 现在考虑满足 $\Delta \Phi_7[17] = 1$ 时需要的条件: $c_2^F[17] = 0$ 并且 $\Delta c_2^F[17] = 0$, 根据前面所总结的 F 函数的性质, 此时 $\Delta \Phi_7[17] = \Delta a_2[17]$, 为了达到这个目标, 必须要求 $\Delta a_2[17] = 1$, 即前面的条件要满足 $a_2[17] = 0, a'_2[17] = 1$ 。

2、 $d_{2,i} = a_{2,i}, d_{2,i} = 0, a_{2,i} = 0$ 确保 c_2^F 的第 i 比特的变化对 Φ_7 不会产生影响, 其中 $i = 1, 2, 4, 5, 25, 27, 29, 30, 31$ 。

根据前面 F 函数的性质, $d_{2,i} = a_{2,i}$ 可以保证 c_2^F 的取值对 Φ_7 没有影响。加上 $\Delta d_{2,i} = 0, \Delta a_{2,i} = 0$ 这两个条件就可以使得 $\Delta \Phi_7[i] = 0$ 。

3、 $c_{2,i} = 1, \Delta c_2[i] = 0$ 确保 a_2 的第 i 比特的变化对 b_2 不会产生影响, 其中 $i = 13, 14, 15, 19, 20, 21, 22, 23$ 。

4、 $\Delta c_2[i] = 1, d_2[i] = a_2[i]$, 其中 $i = 7, 8, 9, 10$, 确保 c_2 的这些比特的变化对 b_2 不会产生影响。

5、 $\Delta c_2[6] = -1, a'_2[6] = 1, d_2[6] = 0, c_2[+28, +29, +30, +31, +32]$, 确保了 c_2^F 的第 28 到第 32 比特的变化不会影响 b_2 。

因为尽管 $c_2[+28, +29, +30, +31, +32]$, 由于 c_2 是 32 比特, 鉴于第 32 比特的特殊性, 可以看成 $c_2[+28, +29, +30, +31, -32]$, 将 c_2 的第 28 比特到第 32 比特联合起来考虑, 根据前面的异或差分和模差分的关系, 可以将这些比特看成 -2^{27} 。为了消除差分, 必须满足 $-2^{27} + (\Delta \Phi_7[6] \lll 22) = 0$ 。条件 $\Delta c_2[6] = -1, a_2^2[6] = 1, d_2[6] = 0$ 使得这个目标得以满足, 从而消除了 c_2^{NF} 第 28 到第 32 比特的差分。

6、 $\Delta c_2[i] = 0, \Delta d_2[i] = 0$, 确保 b_2 的第 3 到 6 比特保持不变, 其中 $i = 13, 14, 15, 16$ 。

7、 $\Delta c_2[i] = 0, \Delta d_2[i] = 0$, 确保 b_2 的第 8 到 12 比特保持不变, 其中 $i = 18, 19, 20, 21, 22$ 。

对第8步输出值 b_2 变化的部分进行分析:

1、 $d_2[11] = 1, b_2[1] = 0, \Delta \Psi_7[11] = 0$, 确保了 b_2 的第1比特的变化。

文献[12]在分析这一比特的部分应该加以修正^[31]。她在文章中这样描述: 条件 $d_2[11] = 1$ 和 $b_2[1] = 0$ 确保 b_2 的第1比特有变化, 具体分析如下:

(1) $d_2[11] = 1, a_2[11] = 0$, 使得 $\Delta \Phi_7[11] = 1$;

(2) 经过移位, $\Delta \Phi_7[11]$ 移位到了第 1 比特;

(3) $\Delta c_2^{NF}[1] = 0, b_2[1] = c_2^{NF}[1] + (\Delta \Phi_7[11] \lll 22) = 0 + 1 = 1$

2、条件

$d_2[26] = 1, a_2[26] = 0, \Delta a_2[26] = 0, \Delta c_2[26] = -1, b_2[16] = 0, b_2[17] = 1$ 确保了 b_2 的第16,17比特的变化。

(1) $\Delta c_2^F[26] = -1$ 使得 $\Delta \Phi_7[26] = a_2'[26] - d_2[26] = -1$;

(2) $\Delta \Psi_7[26] = \Delta \Phi_7[26]$;

(3)

$\Delta b_2[16] = \Delta c_2[16] + (\Delta \Psi_7[26] \lll 22) = \Delta c_2[16] + (\Delta \Phi_7[26] \lll 22) = -1$;

(4) $b_2[16] = 0, b_2[17] = 1$, 单纯考虑 b_2' 的第 16、17 比特, 记为 α, β , 可以很容易得出, $\alpha = 01$, 即 b_2 的第16和17比特发生了变化。

3、条件 $d_2[28] = 0, a_2[28] = 1, d_2[28] = 0, c_2[28] = 1, b_2[21] = 1, b_2[i] = 0$, 其中 $i = 18, 19, 20$, 确保了 b_2 的第18, 19, 20, 21比特的变化。

4、条件 $d_2[3] = 0, a_2[3] = 1, d_2[3] = 0, c_2[3] = 1, b_2[24] = 1$, 确保 b_2 的第24比特的变化。

3.3 消息修改技术

在Hash函数中, 为了降低搜索碰撞所需的明文对的计算复杂度, 必须通过消息修改技术, 把计算机搜索的范围限制有限的范围内。Hash函数中的明文在每轮都是按照不同的顺序使用的。以MD5算法为例, 四轮共64步中用了四组不同顺序的明文: 如果只针对第一轮的消息, 即在前16步的范围内进行修改, 称为单消息修改技术; 如果修改的消息不单纯是第一轮的内容, 即修改的范围跨越了第16步, 则称为多消息修改技术。

3.3.1 单消息修改技术

在MD5算法的前16步(参照文献[13]表格4), 应用了单消息修改技术。以第9步生成 a_3 为例, 为了满足差分路径的需要, 必须满足以下条件:

1、充分条件:

$$a_{3,i} = 1; \quad i = 2, 7, 12, 16, 19, 20, 22, 27, 28, 29, 30$$

$$\begin{aligned} a_{3,j} &= 0; \quad j = 17, 21, 23, 26, 31, 32 \\ a_{3,k} &= b_{2,k}; \quad k = 8, 13, 14, 15 \end{aligned} \quad (13)$$

2、附加条件:

$$\begin{aligned} a_{3,10} &= 0 \\ a_{3,11} &= b_{2,11} \end{aligned} \quad (14)$$

在 a_3 的生成过程中,单消息修改技术是通过将随机产生的 a_3 的值修改成符合上述条件的值,进而“逆推”确立对 a_3 起作用的 m_8 的值。 a_3 的修改过程可以分成三步来实现:

- 1、 $a[3] \wedge (a[3] \& 0\text{xfe7df6e2})$ 这步将所有充分条件和附件条件所涉及的位都清零;
 - 2、 $b[2] \wedge (b[2] \& 0\text{x000074a0})$ 这步提取了 a_3 与 b_2 相关的位;
 - 3、 $10 \times 3\text{c2c8842}$ 这步将 a_3 中必须为1的位置1。
- 通过这三个步骤,可以对MD5算法进行相应的单消息修改。

3.3.2 多消息修改技术

相对于单消息修改,多消息修改技术要复杂得多。多消息修改涉及的范围要比单消息广泛,必须以联系的观点,统筹协调消息的修改和由此引发的对差分路径的影响^[30]。

参照文献[23]的5.1节,本文对自动搜索产生碰撞的明文对的程序进行了分析,为便于说明,下面分8步来介绍多消息修改技术。

第1、2步:应用上节介绍的单消息修改技术,确保 a_2 到 b_4 的充分条件得到满足,进而“逆推”确定 m_6 到 m_{15} 的值,从 m_6 开始确定,是由于文献[13]中明文的差分是从 m_6 开始引入的; m_0 到 m_5 的值没有确定,为满足后面充分条件的需要做好铺垫。

第3步:进入第二轮,随机生成满足充分条件的 a_5 。 a_5 在MD5算法中的表达式: $a_5 = b_4 + \sum_{16} \lll 22$, $\sum_{16} = a_4 + G(b_4, c_4, d_4) + m_1 + 0\text{xf61e2562}$ 。由于上一步可以得知,此时的 m_1 值还未确定;正是由 m_1 值的未确定性来支持 a_5 值的随机产生并且通过修改可以满足充分条件。 m_1 值将在后面的步骤中得以确定。

第4步:按照MD5的算法计算 d_5 , d_5 的表达式中是用了 m_6 。为了满足关于 d_5 的充分条件,由于参与生成 d_5 的其他参数已经确定,只能够修改 m_6 ; m_6 已经在第2步中因 c_2 的充分条件而确定;为了解决这一矛盾,利用在计算 c_2 的表达式: $c_2 = d_2 + \sum_6 \lll 17$, $\sum_6 = c_1 + F(d_2, a_2, b_1) + m_6 + 0\text{xa8304613}$, c_1 是尚未确定的值,利用 c_1 将 m_6 变化的部分“吸收”进去,从而最终确保了 m_6 的变化既不会破坏 c_2 先前的充分条件,又能够满足 d_5 的充分条件。

第5步:目标是利用消息修改技术满足充分条件 $c_{5,27} = d_{5,27}$, c_5 的表达式为 $c_5 = d_5 + \sum_{18} \lll 14$, $\sum_{18} = c_4 + G(d_5, a_5, b_4) + m_{11} + 0 \times 265e5a51$ 。

1、为了满足充分条件,必须关注 $\sum_{18,13}$,进而关注 $m_{11,13}$,使其的变化引起 $\sum_{18,13}$ 的变化,从而实现充分条件 $c_{5,27} = d_{5,27}$ 。

2、联系第一次用到 m_{11} 的地方: $b_3 = c_3 + (b_2 + F(c_3, d_3, a_3) + m_{11} + 0 \times 265e5a51) \lll 22$;在不影响 b_2 和 a_3 的涉及充分条件的关键位的前提下,通过修改 $b_{2,13}$ 和 $a_{3,13}$ 来促使 $m_{11,13}$ 的变化。

3、由于加法进位的原因, $b_{2,13}$ 和 $a_{3,13}$ 的修改也将确保 $c_{5,32} = 0$ 的充分条件的满足。

4、 a_3 , b_2 的变化将分别对其下面的四步MD5运算产生影响。在第1步已经确定了前16步产生的链结变量的充分条件。为了不影响第1步确定的充分条件,必须对 m_8 , m_9 , m_{10} , m_{11} , m_{12} 进行修改,这些工作将在第6~7步完成。

第6步:目标是用明文修改技术确保 b_5 的充分条件的满足。

1、同第3步产生 a_5 的原理一样,随机产生满足充分条件的 b_5 ,此时与 b_5 产生过程相关联的是尚未确定值的 m_0 , b_5 产生后确定了 m_0 的值;

2、第3步已经定了 a_5 的值,从而 m_1 的值得以确立,通过MD5算法,计算 a_1 和 d_1 ,进而确定了第2步尚未确定的 m_2 , m_3 , m_4 , m_5 的值。这几个值的最后确定,为整体上充分条件的满足创造了很大的消息修改的空间,降低了计算复杂度。

第7步:应用上述所介绍的技术来满足 a_6 的充分条件。此处的附加条件必须结合前面介绍的非线性函数的性质,控制消息修改技术对链结变量的影响范围,减少不必要的修改。

第8步:利用计算机的随机数产生和程序中设计的自动搜索功能,在充分条件满足的范围内进行有限制的搜索能够产生碰撞的明文。

3.3.3 程序实现

对MD5进行成功碰撞攻击的目标,是设法得到产生相同hash值的一对明文。利用消息修改技术,可以借助于计算机的程序自动搜索的能力,在确定好差分的引入和设计好差分路径的前提下,有效地降低计算复杂度,提高成功搜索到明文的概率。

根据上面对消息修改技术的解析,在文献[14]对其条件进行了进一步的如下完善;

1、在确定 a_2 的约束条件时,增加了: $a_{2,7} = b_{1,7}$

- 2、在确定 c_2 的约束条件时,将后面附加条件提到前面实现, 增加: $c_{2,7} = d_{2,7}$
 - 3、 a_3 的条件增加: $a_{3,11} = b_{2,11}, a_{3,10} = 0$, 以取代后面对 d_3 条件 $\Sigma_{9,31}$ 的判断。
 - 4、增加对 b_3 的限制,确保实现 $b_{3,31} = 0 \rightarrow \Sigma_{12,25} = 0$
- 第一块明文的搜索程序如图3.1所示:

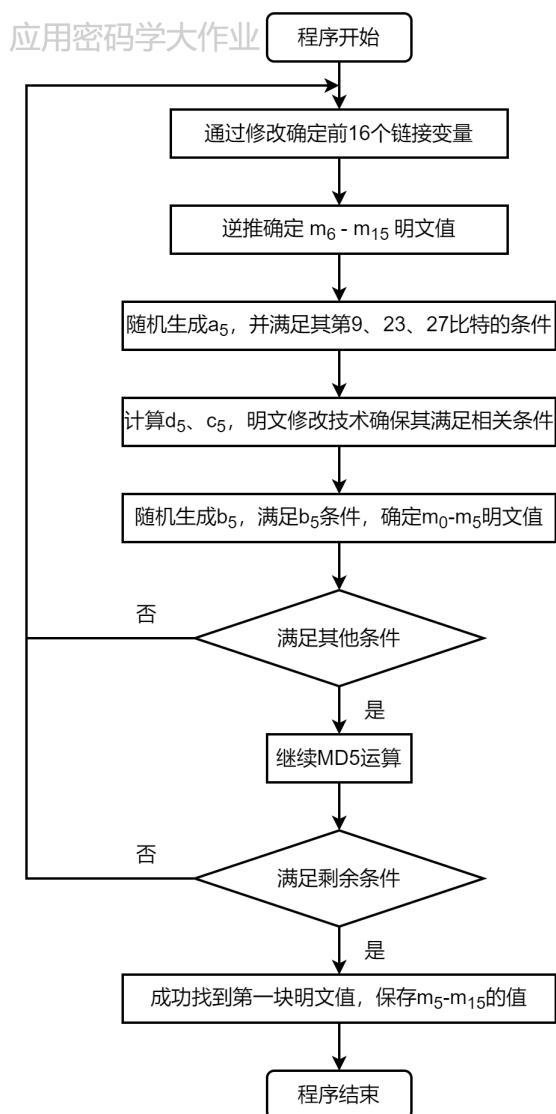


图3-1 第一块明文搜索流程

目前针对MD5的碰撞攻击,都是采用消息长度为两个明文块的输入,最后实现碰撞。通过消息修改技术对第一个明文块进行搜索是整个搜索的关键,整个碰撞明文块搜索的计算复杂度也主要集中在第一块上。搜索的程序主要采用的是 *if - then - else* 的结构,对限制条件进行分析判断,产生如下两种情况:

- 1、满足所要求的条件,程序继续进行,执行之后的MD5的步骤

2、不满足所要求的条件,对前面影响该值的步骤或者链结变量进行局部微调;如果局部调整不起作用,回到最初,重新随机产生明文值,直到满足所有条件为止。对程序条件修改的目的主要是针对第2种情况,如果一旦运行的程序因不满足判断条件而回到了程序的开始,那么整个搜索的计算复杂度就比较高。将基本的条件提到比较靠前的位置实现,对于下面程序条件的满足至关重要。

3.3.4 消息修改技术总结

单消息修改技术仅用于MD5算法的第一轮,其方法是通过先确定16个链接变量的值,再反过来确定相对应的16个明文字的值,是个逆推的过程;其特点是16个明文字一步修改到位,确定性强。

多消息修改技术主要应用于MD5算法第16步以后的运算,其中包含单消息修改技术。在应用中,首先要确保前面步骤已经满足的充分条件在之后的消息修改中不被间接修改;其次是确保逆推的过程和正推的过程在某一步骤中可以顺利衔接起来;最后尽可能限制计算机搜索范围,有效降低计算复杂度。其特点是必须借助于计算机程序自动搜索功能,具有一定的随机性。

3.3.5 碰撞设计的基本原则

1、产生碰撞是研究的目标。同样的差分引入,可能产生许多种不同的差分路径。有些路径通过控制是在理论上是可能产生碰撞的,称之为“可行的差分路径”;有些路径是不可能产生碰撞的,称之为“不可行的差分路径”。针对MD结构,为了达到这个目标,最关键的是引入适当的差分并在MD5算法迭代中寻找“可行的差分路径”,按照最终的需要对差分路径进行有效的控制:模差分 and 符号差分灵活转换,符号差分可以按照需要进行扩张,不必要的差分在迭代中消除。

2、在MD5碰撞中,整个运算过程中的差分包括两部分:引入的差分和迭代运算中的差分,这两部分在本文中统称为:差分。一个好的差分首先要确保从第二轮迭代运算开始差分越来越少;其次,好的差分在第一次引入时要远离MD5运算的第1步,这样在消息修改技术的应用中,可以获得更多的“自由字”,降低消息修改的计算复杂度,提高碰撞的概率。

3、由于原则2的要求,第16步一般要出现局部碰撞,即在第16步的差分为0,使得进入第二轮以后差分能够控制在比较小的范围内。所以在差分路径控制过程中,将第16步作为一个承上启下的基点,这在尝试寻找“可行的差分路径”时至关重要。第二轮的差分小,一般在第二轮的后半部分,进入了稳定的局部碰撞阶段。

4、尽量将差分路径引到第32比特 (MSB)。在差分中,第32比特是特殊的,其“+”、“-”号在某种程度上可以看成是一致的。第32比特的差分很多情况下,都可以以抵消或者溢出的形式将该比特位的差分消除。差分在进入第三轮时,一般都只出现在第32比特。这点对于第一明文块3比特差分来说,为了在第三轮都保

持 MSB,明文引入差分时一般要让某一比特差分移位后恰好移动到MSB,这是在差分引入中心须考虑和权衡的。

5、布尔函数的性质的应用在差分路径的控制中起着重要的作用,其中 **F** 函数和 **H** 函数用的最多。很多情况下,利用 **F** 函数的性质来进行差分的扩展和消除,利用 **H** 函数将迭代中的差分保持在 **MSB**。布尔函数性质的应用技巧很多体现在第一和第三轮,由此也是确立充分条件的关键。

6、目前的碰撞都是两明文块产生的。在依靠消息修改技术的基础上,重点是放在第一明文块的搜索上。在文献 [13] 第二明文块差分的引入和第一明文块呈严格的对称性;文献 [12] 突破了这种对称模式,是个进步;但是对于差分路径的控制,三组碰撞还是在第一明文块和第二明文块之间采取比较对称的形式。这种对称结构有个比较明显的优点:可以靠抵消或者溢出最终实现碰撞的目标。

3.3.6 碰撞攻击的评价

通过三组已经发表的碰撞数据,对某些指标进行量化比较,一个碰撞攻击可以从以下八个方面进行评价:

- 1、该攻击是否依赖于固定的初始值。
- 2、产生碰撞所需要的明文的块数。
- 3、在消息修改技术中,“自由字”个数的多少。
- 4、整个明文块中差分引入的比特数。
- 5、碰撞所需要的充分条件的个数。
- 6、碰撞所需要的附加可能条件的个数。
- 7、寻找产生碰撞的明文对的平均计算复杂度。
- 8、在相同配置的计算机中,找到碰撞明文对平时用时。

第四章 项目功能介绍

4.1 环境配置

4.1.1 代码下载与编译

本项目代码已在github中开源。仓库地址为<https://github.com/coronaPolvo/fastcoll>。用户可以执行如下指令下载全部仓库：

```
git clone https://github.com/coronaPolvo/fastcoll.git
```

本项目需要在Linux下进行编译运行。在项目仓库中已提供好Makefile文件，用户可以通过以下命令编译运行：

```
make
```

4.1.2 boost环境配置

在代码实现中使用了 C++ boost库因此需要配置boost的使用环境。配置步骤如下：

1. 在官网下载boost后解压
2. 进入到boost文件夹中以root权限运行./bootstrap.sh
3. 运行./b2 install

4.2 功能概述

本项目主要功能为使用构造前缀碰撞法来实现MD5的碰撞。给定任意格式的一个前缀，

4.3 任意文件MD5碰撞

在本小节中将对.txt、.py以及编译cpp文件生成的可执行文件进行md5碰撞测试，并展示碰撞结果。首先创建一个txt文本，接着使用fastcoll进行MD5碰撞。运行结果如图4.1所示。

```
keter — fish /home/ray/workspace/coronapolvo/fastcoll — ssh ray@10.0.0.5 — 80x21
~ — fish /home/ray/workspace/coronapolvo/fastcoll — ssh ray@10.0.0.5

→ echo "123456" > test.txt (base)
ProfessorRay > w > c > fastcoll master
→ ./fastcoll test.txt -o out1.txt out2.txt (base)
MD5 collision generator
应用密码学大作业-任意文件MD5碰撞

Using output filenames: 'out1.txt' and 'out2.txt'
Using prefixfile: 'test.txt'
Using initial value: 7397deb246e347915944fd83aa27b8f9

Generating first block: .....
Generating second block: S10.....
Running time: 13.2794 s
ProfessorRay > w > c > fastcoll master
→ cat out1.txt (base)
123456
?H螺0???Ĉ???#???T#???g?_h??Sh4??TvJ;\???d??I????_??_??

ProfessorRay > w > c > fastcoll master
→ cat out2.txt (base)
123456
?H螺0???Ĉ???#???3T#???g?_h??Sh4??TvJ;\???d??I???z?_??_??
```

图4.1 txt文本碰撞

接下来尝试对python代码文件进行碰撞, 首先使用如下代码进行代码碰撞。

```
In:
echo "print(1111)" > a.py
./fastcoll a.py -o out1.py out2.py

Out:
MD5 collision generator
应用密码学大作业-任意文件MD5碰撞

Using output filenames: 'out1.py' and 'out2.py'
Using prefixfile: 'a.py'
Using initial value: ee2dff34a0695e52214b1bb5092afd2c

Generating first block: .....
Generating second block: S01.....
Running time: 6.423 s
```

接下来让对生成的python文件的md5进行验证:

```
In: md5sum out1.py
Out:a01d2ce72bf21e8c8ba5e41d1ac4fc2d  out1.py
```

```
In:md5sum out2.py
Out:a01d2ce72bf21e8c8ba5e41d1ac4fc2d  out2.py
```

通过运行输出的碰撞文件对python文件的可用性进行验证:

```
In: python3 out1.py
Out:1111
In: python3 out2.py
Out:1111
```

使用cat命令查看两个问价的内容:

```
In: cat out1.py
Out:
print(1111)
h?tU8U;%Ć?~j*?????:?6?2????n??o&j 'ç+???c???!W`??S????zTo??\
<0!??L??=!?=??Zv??G?kVd>????th????n?i???)1}??2yE?3

In: cat out2.py
Out:
print(1111)
h?tU8U;%Ć?~j*???g????:?6?2????n??o&j 'R?+???c???!W`?8??S????zTo??\
<0!??L?Ž!?=??Zv??G?kVd>????th??x?n?i???)1}??yE?3
```

接下来尝试对g++编译对可执行文件进行测试, 首先编写如下的c++代码:

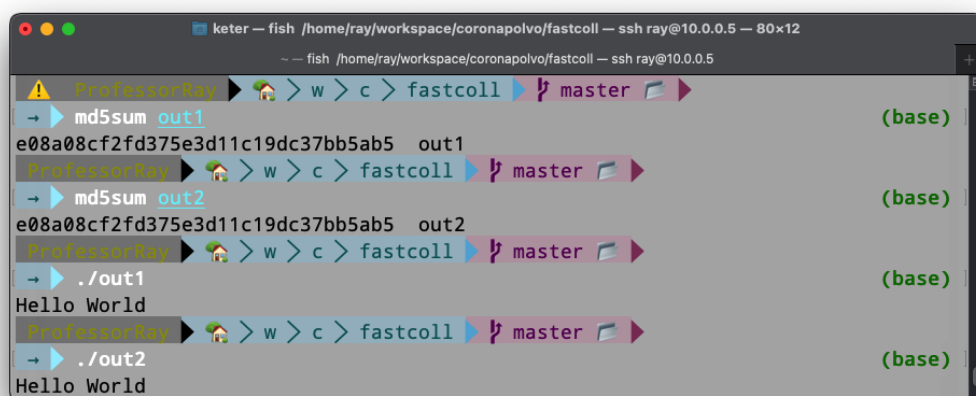
```
#include <iostream>
using namespace std;

int main(){
    cout << "Hello World" << endl;
    return 0;
}
```

使用g++对上面的文件进行编译并测试代码运行结果:

```
In:
g++ a.cpp -o a
./a
Out:
Hello World
```

使用fastcoll对a进行碰撞后测试生成的文件, 测试结果如下图4.2所示:



```
keter — fish /home/ray/workspace/coronapolver/fastcoll — ssh ray@10.0.0.5 — 80x12
— fish /home/ray/workspace/coronapolver/fastcoll — ssh ray@10.0.0.5
ProfessorRay > w > c > fastcoll master (base)
→ md5sum out1
e08a08cf2fd375e3d11c19dc37bb5ab5 out1
ProfessorRay > w > c > fastcoll master (base)
→ md5sum out2
e08a08cf2fd375e3d11c19dc37bb5ab5 out2
ProfessorRay > w > c > fastcoll master (base)
→ ./out1
Hello World
ProfessorRay > w > c > fastcoll master (base)
→ ./out2
Hello World
```

图4.2 碰撞测试结果

4. 4 项目在CTF中的运用

4. 4. 1 强网杯CTF比赛

强网杯采取在线解题模式（Jeopardy模式），主要面向国内高等院校和网络安全企业、机构。赛题内容主要涉及二进制程序逆向分析、密码分析、智能终端安全、信息隐藏、人工智能等网络安全领域的主要知识与技能，重点考察参赛人员网络安全知识的综合运用能力和网络安全技能的创新实践能力。

4. 4. 2 强网杯中的MD5碰撞

在第二届强网杯中有一道题目如下，该题目主要考察PHP语言环境下绕过MD5碰撞检验。

```
if($_POST['param1']!==$_POST['param2'] &&
md5($_POST['param1'])===md5($_POST['param2']))){
    die("success!");
}
```

题目中两次的比较均用了严格的比较，无法通过弱类型的比较去绕过。那么就可以考虑在短时间内构造两个不一样的字符串。

4. 4. 3 构造

创建一个文本文件。写入1。命名为init.txt。运行fastcoll 输入以下参数。-p 是源文件 -o 是输出文件。运行几秒钟以后文件就生成好了。

4. 4. 4 测试

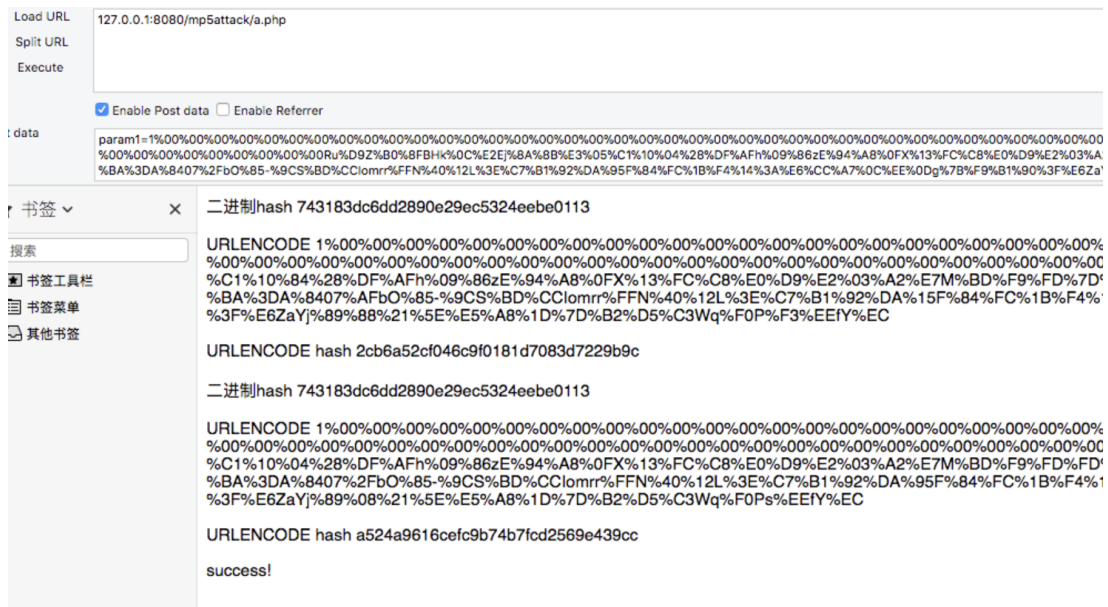


图4.4 运行结果

第五章 总结

本论文主要对Hash函数MD5进行了解析和研究以及通过复现代码的方式实现了任意文件的MD5碰撞。MD5算法做为Hash函数中的一员,是MD结构的典型代表,是一种比较成熟的设计思想。通过对MD5算法的研究,可以进一步掌握Hash函数有效的分析方法,这对将来新的Hash函数SHA-3的分析和研究有着重要的借鉴意义。

参考文献

- [1] RIVEST RL, The MD4 message digest algorithm. Advances in Cryptology-crypto 90, Springer-verlag 1991, LNCS 537: 303-311
- [2] RIVEST RL, The MD5 message-digest algorithm. Request for Comments RFC 1320,1992
- [3] ZHENG Y, PIEPRZYK J SEBERRY. HAVAL-A one-way Hashing algorithm with variable length of output. Advances in Cryptology, Auscrypto'92, LNCS 718 83-104
- [4] RIPE. Integrity primitives for secure information systems. Final report of RACE integrity primitives evaluation(RIPE-RACE 1040), LNCS 1007, 1995
- [5] FIPS 180: Secure Hash standard, Federal Information Processing Standards Publication. NIST. 1993
- [6] FIPS 180-1. Secure Hash standard, Federal Information Processing Standards Publication. NIST. 1995
- [7] Xiaoyun W, Xuejia Lai, Dengguo Feng. Cryptanal ysis of the Hash Functions MD4
- [8] Xiaoyun W, Hongbo Yu. How to break MD5 and other Hash Functions EUROCRYPT 2005. LNCS 3494: 19-35
- [9] 王小云,冯登国,于秀源,中国科学,信息科学2005,35(3):1-12
- [10] XIE T, FENG D G. A new differential for MD5 with its full differential path 2009-04-10. <http://eprint.iacr.org/2008/230.ndi>
- [11]XIE T, LIU F B. Could I-MSB input differential be the fastest collision attack for MD52009-04-20. <http://eprint.iacr.org/2008/391> .
- [12] XE T, LIU F B. Could I-MSB input differential be the fastest collision attack for MD52009-04-20. <http://eprint.iacr.org/2008/391>
- [13] Xiaoyun W, Hongbo Yu. How to break MD5 and other Hash Functions EUROCRYPT2005 LNCS 3494: 19-35
- [14] Marc Stevens, Arjen K. Lenstra, Benne de Weger. Chosen-prefix collisions for MD5 and applications. 2012.

