关于 Hash 函数 MD5 的解析

毛 明[1,2] 陈少晖[1,2] 袁 征[2,3] 贾永兴[1,2]
(西安电子科技大学通信工程学院 西安 710071)[1] (北京电子科技学院信息安全系 北京 100070)[2]
(清华大学高等研究中心 北京 100084)[3]

摘 要 在王小云写的关于分析 MD5 算法文章的基础上,对 MD5 算法的破译进行进一步的解析。以 MD5 算法第八步为例,介绍了 F 函数的性质和差分路径的有效控制。从手工推算和程序实现两方面对 MD5 算法进行了解析和测试,对王的文章进行了部分修正,进一步说明了满足差分特征的条件和关键点。这对 MD5 和其他 Hash 函数的分析破译有着重要的作用。

关键词 MD5,差分控制,Hash 函数,解析
中图法分类号 TP309 **文献标识码** A

Cryptanalysis of the Hash Function MD5

MAO Ming[1,2] CHEN Shao-hui[1,2] YUAN Zheng[2,3] JIA Yong-xing[1,2]
(Department of Communication Engineering, Xidian University, Xi'an 710071, China)[1]
(Department of Information Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China)[2]
(Center for Advanced Study, Tsinghua University, Beijing 100084, China)[3]

Abstract This paper is about the cryptanalysis of the Hash function MD5, on the basis of the paper concerning the breaking the MD5 which was written by Wang et al. In this paper, the eighth step of MD5 algorithm was taken for example and the properties of the F function and the effective control of the differential path were introduced. The MD5 algorithm was cryptanalyzed from both the manual calculation and the testing program and some part of the paper written by Wang was amended. Finally, the conditions and key points of meeting the differential characteristics were discussed. In general, this paper plays an important role in cryptanalyzing and breaking the MD5 algorithm.

Keywords MD5, Differential control, Hash function, Cryptanalysis

随着信息安全技术的发展,Hash 函数的重要性日益提高,它在公钥密码、数字签名、完整性检验、身份认证等领域中有着广泛的应用。因此,人们也对其进行了深入的研究。

Hash 函数(杂凑函数),就是能够把任意有限长的消息串 M 映衬成为某一固定长度的输出串 h 的一种函数。这个输出串 h 称为消息串 M 的消息摘要。消息摘要可以看作是数字指纹,像现实生活中指纹可以验证一个人身份的唯一性一样,Hash 函数也可用于验证一个消息的唯一性。正是由于这个特性,Hash 函数在信息安全的认证领域中得到广泛应用。Hash 函数主要有 MDx 系列和 SHA 系列,MDx 系列包括 MD4[1],MD5[2],HAVAL[3],RIPEND[4] 等;SHA 系列包括 SHA-0,SHA-1[5],SHA-256,SHA-384[6] 等,这些 Hash 算法体现了目前主要的 Hash 函数设计技术。

在 2004 年国际密码年会上,王小云等中国研究人员公布了一种破解 MD4[7],MD5[8],SHA-0[9],SHA-1[10],HAVAL-128[11],RIPEMD 等 Hash 函数的方法,其攻击算法复杂度都大大降低,且在一般的个人电脑上运算就可以找到相关碰撞的实例。这就对现有的 Hash 函数提出了严峻的挑战,促进了新的 Hash 算法的开发研究。

本文以王小云写的 MD5 文章为基础,对 MD5 算法的破译进行了解析,对 F 函数的性质和差分特征进行了详细的分析。

1 MD5 算法

MD5 算法是 MD4 算法的改进算法。Ron Rivest 于 1990 年提出 MD4 单向散列函数,MD 表示消息摘要(Message Digest),对输入消息,算法产生 128 位散列值。该算法首次公布之后,Ben den Boer 和 Antoon Bosselaers 对算法三轮中的后两轮进行了成功的密码分析。在一个不相关的分析结果中,Ralph MerKle 成功地攻击了前两轮。尽管这些攻击都没有扩展到整个算法,但还是暴露了 MD4 算法的不足,因此 Rivest 对 MD4 算法进行了改进,MD5 算法就此产生。

到稿日期:2008-12-24 返修日期:2009-03-09 本文受国家 973 计划(编号:2007CB807902),中国博士后科学基金资助项目(编号:20080430423)资助。

毛 明(1963—),男,教授,硕士生导师,主要研究方向为信息安全,E-mail:chensh@mail.besti.edu.cn;陈少晖(1983—),男,硕士生,主要研究方向为信息安全和密码学;袁 征(1968—),女,副教授,硕士生导师,主要研究方向为信息安全和密码学;贾永兴(1982—),男,硕士生,主要研究方向为密码通信技术。

MD5 算法可以简要地叙述为：MD5 以 512bit 分组来处理输入的信息，且每一分组又被划分为 16 个 32bit 子分组，经过了 4 轮的压缩处理后，算法的输出由 4 个 32bit 分组组成，将这 4 个 32bit 分组级联后将生成一个 128bit 消息摘要。

2 预备知识

2.1 F 函数的性质

MD5 第一轮使用的逻辑函数是 $F(X,Y,Z)=(X\wedge Y)\vee(\neg X\wedge Y)$。

从 $F(X,Y,Z)$ 函数的表达式，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，可以总结出 F 函数的 3 条性质[7]：

(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 $X=1$。

这 3 条性质对于研究 MD5 第一轮压缩函数有着重要的作用，利用这 3 条性质，可以控制差分路径，有利于后面的差分分析和碰撞的寻找。

2.2 异或差分和模差分

异或差分[8]：设 X 和 X' 是两个 32bit 的数，异或差分将两个数每一 bit 的不同都体现出来，异或差分的值为 $X\odot X'$。为了更好地体现这两个数的某一 bit 的差别，在异或差分的基础上进一步引入了“+”、“−”号。例如，$X'-X=[5,6,7,8\cdots-27]$，这说明 X 和 X' 从第 5 到第 27bit 都不同，而且从符号可以看出 X 的第 5 到第 26bit 是 0，第 27bit 是 1；而 X' 的第 5 到第 26bit 是 1，第 27bit 是 0。这种带符号的异或差分可以很容易看出两个数的各 bit 位的差别。

模差分[8]：设 X 和 X' 是两个 32bit 的数，模差分将两个数的差别用整数减法体现出来，同样以上面的为例，$X'-X$ 用异或差分表示为 $[5,6,7,8\cdots-27]$，此时用模差分表示为 $X'-X=-2^4$。

在分析过程中，将同时使用两种差分方法来表示两个数的差分。这样可以更好地从单个 bit 的变化和整体数值两方面体现两个数的差分，下面将以具体的例子来说明两种差分同时使用的好处。

模差分 $X'-X=2^4$，可以有以下几种不同的异或差分：

i. 1bit 不同：$X\odot X'=$0x00000010，即 X' 的第 5bit 是 1 而 X 的第 5bit 是 0；

ii. 2bit 不同：$X\odot X'=$0x00000030，即 X' 的第 6bit 是 1，第 5bit 是 0 而 X 的第 6bit 是 0，第 5bit 是 1；

iii. 3bit 不同：$X\odot X'=$0x00000070，即 X' 的第 7bit 是 1，第 5、第 6bit 是 0 而 X 的第 7bit 是 0，第 5、第 6bit 是 1；

iv. 4bit 不同：$X\odot X'=$0x000000f0，即 X' 的第 8bit 是 1，第 5、第 6、第 7bit 是 0 而 X 的第 8bit 是 0，第 5、第 6、第 7bit 是 1；

v. 同样的道理，这样可以有更多的借位，最多的可以一直到第 32 位不同，即 $X\odot X'=$0xfffffff0。

2.3 控制差分路径时应注意的问题

就 MD5 的第一轮而言，以 4 个连续步骤为例：

$a=b+((a+F(b,c,d)+m_i+t_i)<<<s_i)$；

$d=a+((d+F(a,b,c)+m_{i+1}+t_{i+1})<<<s_{i+1})$；

$c=d+((c+F(d,a,b)+m_{i+2}+t_{i+2})<<<s_{i+2})$；

$b=c+((b+F(c,d,a)+m_{i+3}+t_{i+3})<<<s_{i+3})$；

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 3 个参数某一 bit 的变化，最终只会影响 F 函数的相应的 bit 位；F 函数以外的运算，由于可能存在进位的问题，因此影响的可能不只是变化的那一 bit，还可能影响其周围的一连串的 bit 位，影响的范围可能很大。在整个运算过程中，第 32bit 位置特殊，其进位将会产生溢出，这个性质在差分分析过程中至关重要。

3 对 MD5 第 8 步的分析

结合上面的预备知识，对 MD5 的第 8 步进行分析，介绍了 F 函数性质在差分分析中的应用和差分路径的控制。

3.1 符号说明

1. $M=(m_0,m_1\cdots,m_{15})$ 和 $M'=(m_0',m_1'\cdots,m_{15}')$ 代表两个 512bit 的明文块，$\Delta M=(\Delta m_0,\Delta m_1\cdots,\Delta m_{15})$ 代表两个明文块的差分，其中每个 m_i 为 32bit 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 $1\leqslant i\leqslant 16$；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 bit，其中，最高位为第 32bit，最低位为第 1bit。

4. $\Phi_{i,j},\Psi_{i,j}$ 分别表示 Φ_i 和 Ψ_i 的第 j bit。

5. $\Delta x_{i,j}=x_{i,j}'-x_{i,j}=\pm 1$，这是表示第 j bit 的差分。如果值为 1，说明 x_i' 的第 j bit 是 1 而 x_i 的第 j bit 是 0；如果值是 0，说明 x_i' 的第 j bit 是 0 而 x_i 的第 j bit 是 1，其中 x 可以是 a,b,c,d,Φ,Ψ。

6. $\Delta x_i[j_1,j_2,\cdots,j_l]=x_i[j_1,j_2,\cdots,j_l]-x_i$，表示 x_i 的第 $j_1,j_2,\cdots,j_l$ bit 的变化。$x_i[\pm j_1,\pm j_2,\cdots,\pm j_l]$ 表示具体第 $j_1,j_2\cdots j_l$ 的 bit 的变化，其中“+”表示该 bit 从 0 变成了 1；“−”表示该 bit 从 1 变成了 0。

3.2 分析过程

王的文章[8]是用两个明文块 (M_0,M_1)，明文长度共为 1024bit，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，这意味着成功找到了碰撞。本文只对第一次 MD5 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，本文结合王的文章[8]，以第 8 步为例，介绍 F 函数的性质的运用和差分路径的控制。

第 8 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，满足 $b_2'=b_2[1,16,-17,18,19,20,-21,-24]$ 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。为了达到第 8 步的差分目标，必须确保以下前提条件：

$b_1'=b_1$；

$a_2'=a_2[7,\cdots 22,-23]$；

$d_2'=d_2[-7,24,32]$;

$c_2=c_2[1,2,3,4,5,-6,7,8,9,10,11,-12,-24,-25,-26,27,28,29,30,31,32]$

第 8 步的运算如下：

$b_2=c_2^{NF}+((b_1+F(c_2^F,d_2,a_2)+m_7+t_7)<<<22)$

$b_2'=c_2^{NF}'+((b_1'+F(c_2^F',d_2',a_2')+m_7'+t_7')<<<22)$

b_2 的表达式中有两个 c_2，为了更好区分，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 $t_7'=t_7$；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'。

3.2.1 *应用 F 函数的性质，对第 8 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 bit 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 3 条性质：$F(X,Y,Z)=F(X,Y,\neg Z)$，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 bit 的变化不会影响 b_2，其中 $i=13,14,15,16,17,18,19,20,21,22,23$。

3.2.2 *对第 8 步的输出 b_2 不变化的部分进行分析*

(1)$c_2[17]=0,\Delta a_2[17]=1$ 确保 b_2 的第 7bit 没有变化。

c_2^{NF} 的第 7 到第 12bit 的变化，按照前面的预备知识，可以将其统一起来考虑，尽管它的异或差分有 6bit，但是它的模差分就只有 -2^6。$\Delta b_2[7]=\Delta c_2^{NF}[7,\cdots,11,-12]+(\Delta\Psi_7[17]<<<22)=-2^6+(\Delta\Psi_7[17]<<<22)$。为了保持 b_2 的第 7bit 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 bit。这需要运用明文的修改技术，这不是本文的重点，不予过多讨论。根据王的文章[8]表 2 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)$d_{2,i}=a_{2,i},\Delta d_{2,i}=0,\Delta a_{2,i}=0$ 确保 c_2^F 的第 i bit 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 bit 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 bit 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 28bit 到第 32bit 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 32bit，鉴于第 32bit 的特殊性，可以假设虚拟的第 33bit 为 $c_2[-33]$，将 c_2 的第 28bit 到虚拟的第 33bit 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 bit 看成 -2^{27}。为了消除差分，必须满足 $-2^{27}+(\Delta\Phi_7[6]<<<22)=0$。条件 $\Delta c_2[6]=-1,a_2'[6]=1,d_2[6]=0$ 使得这个目标得以满足，从而消除了 c_2^{NF} 第 28bit 到第 32bit 的差分。

(6)$\Delta c_2[i]=0,\Delta d_2[i]=0$，确保 b_2 的第 3bit 到第 6bit 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 8bit 到第 12bit 保持不变，其中 $i=18,19,20,21,22$。

3.2.3 *对第 8 步的输出 b_2 变化的部分进行分析*

(1)$d_2[11]=1$ 和 $b_2[1]=0,\Delta\Psi_7[11]=0$，确保了 b_2 的第 1 bit 的变化。

王的文章[8]在分析这一 bit 的部分应该加以修正。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 1bit 有变化，具体分析如下：

i. $d_2[11]=1,a_2[11]=0$，使得 $\Delta\Phi_7[11]=1$；

ii. 经过移位，$\Delta\Phi_7[11]$ 移位到了第 1bit；

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

该分析过程中有两处需要修正：从王的文章[8]附录的表 3，可以很容易看出 $\Delta c_2^{NF}[1]=1$，所以如果按照 $\Delta b_2[1]=\Delta c_2^{NF}[1]+(\Delta\Phi_7[11]<<<22)=1+1=0$(进位)，这就不能够满足 b_2 第 1 bit 变化的要求。其实，单纯考虑 $\Delta\Phi_7[11]$ 是不全面的。正确的表达式为 $\Delta b_2[1]=\Delta c_2^{NF}[1]+(\Delta\Psi_7[11]<<<22)$。第 8 步整体上满足 $\Delta\Psi_7=\Delta\Phi_7$，但是从某一 bit 位，不一定满足 $\Delta\Psi_7[i]=\Delta\Phi_7[i]$。原因是由于在 Ψ_7 的表达式中，只有 F 函数的运算是位运算，其他的都是在模 2^{32} 下的运算，这样就可能产生进位扩散的问题。正是由于进位扩散的问题，$\Delta\Psi_7[11]\neq\Delta\Phi_7[11]$(根据王的文章[8]表 2 提供的数据程序进行测试，得到该结果)。此处，$\Delta\Phi_7[11]=1,\Delta\Psi_7[11]=0$，所以 $\Delta b_2[1]=\Delta c_2^{NF}[1]+(\Delta\Psi_7[11]<<<22)=1+0=1$。

(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 16bit 和 17bit 的变化。

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

ii. $\Delta\Psi_7[26]=\Delta\Phi_7[26]$；

iii. $\Delta b_2[16]=\Delta c_2[16]+(\Delta\Psi_7[26]<<<22)=\Delta c_2[16]+(\Delta\Phi_7[26]<<<22)=-1$

iv. $b_2[16]=0,b_2[17]=1$，单纯考虑 b_2' 的第 16bit 和 17bit，记为??，? $-10=-1$，可以很容易得出，?? $=01$，即 b_2 的第 16bit 和 17bit 发生了变化。

(3)条件 $d_2[28]=0,a_2[28]=1,\Delta d_2[28]=0,\Delta c_2[28]=1,b_2[21]=1,b_2[i]=0$，其中 $i=18,19,20$，确保了 b_2 的第 18bit，第 19bit，第 20bit，第 21bit 的变化。

(4)条件 $d_2[3]=0,a_2[3]=1,\Delta d_2[3]=0,\Delta c_2[3]=1,b_2[24]=1$，确保了 b_2 的第 24bit 的变化。

3.2.4 *对第 8 步输出差分特征的小结*

b_2 的差分总体为表达式 $\Delta b_2=\Delta c_2^{NF}+(\Delta\Psi_7<<<22)$

(1)对于 $\Delta b_2[i]$，即 b_2 的第 i bit 的变化，关注 $\Delta c_2^{NF}[i]$ 和 $\Delta\Psi_7[k]$ 的情况，此时 $i=(k+22)\%32$。

(2)对于 $\Delta c_2^{NF}[i]$，关注第 i bit 前后 bit 的关系，以联系的

(下转第 164 页)

处理机数为2～4时的并行程序保持较为理想的加速比，而当处理机数为8时加速比值较差。分析其原因，在于该并行程序的性能主要受到内存系统性能的限制，而非CPU的数量或CPU性能。对比表1中显式方法与隐式方法的加速比，随着处理机数的增加，隐式方法要好于显式方法。其原因一方面在于当每个处理机所分配的问题区域减小的同时，每个区域的对角部分也相应减小，因此求解块状雅克比预条件子所引入的计算量将减少。另一方面由于ILU不完全分解方法仅对位于对角线上的带状区域进行计算，随着对角线上的带状区域变得越来越窄，其计算量也将减少。对比表1中显式方法和隐式方法下的执行时间，前者要远低于后者。这是因为显式算法在进行迭代时仅用到邻近网格点的离散信息更新解向量，无需使用全局的线性或非线性解，而隐式方法每一步迭代需在全局范围内传播信息。因此，当时间步长可满足显式格式收敛性条件时，显式方法的程序执行速率将明显优于隐式方法。

实验二：取网格划分为100 * 100 * 8～100 * 100 * 128，表2中分别给出了由4个节点组成的并行系统在不同问题规模下的相对加速比和计算效率。

表2　不同问题规模下的相对加速比和计算效率

Grids	Escaping Time (s)	Speedup	Efficiency
100 * 100 * 8	45.17	2.66	66.58%
100 * 100 * 16	79.09	2.92	73.03%
100 * 100 * 32	146.72	2.90	72.37%
100 * 100 * 64	286.50	2.99	74.84%
100 * 100 * 128	591.95	2.95	73.73%

从表2可看出，当问题规模为100 * 100 * 8的网格时，并行程序的计算效率仅为66.58%，而问题规模扩大后效率基本稳定在73%～74%。分析其原因，一方面在于随着问题规模的扩大，计算初始化以及输入、输出所占的时间比重将减小，并行程序将获得更高的计算效率。另一方面，当问题规模达到一定程度，单机在计算取数据过程中存在大量的内外存交换开销，而并行系统大多数数据处理操作都是并行地在本地内存中执行，通信开销和计算开销均比较小。

结束语　热传导方程在地下水流动数值模拟、油藏数值模拟等工程计算中有着广泛应用，其并行实现是加快问题求解速度、提高问题求解规模的重要手段。然而，由于高精度模拟中所涉及的数据存储和计算等所带来的复杂性常常使得这类问题难于高效求解。本文提出的热传导偏微分方程求解策略采用了分布式内存技术来解决超大规模稀疏矩阵的存储以及计算问题，整合了各种高效并行迭代解法。实验表明，基于该策略所实现的面向大规模热传导问题并行求解程序具有良好的并行性能以及可扩展性。

参考文献

[1] Dawson C N, Du Qiang, Dupont T F. A Finite Difference Domain Decomposition Algorithm for Numerical solution of the Heat Equation[J]. Mathematics of Computation, 1991, 195(57):63-71

[2] Zhang Bao-lin, Wan Zheng-su. New techniques in designing finite-difference domain decomposition algorithm for the heat equation[J]. Computers & Mathematics with Applications, 2003, 45(10/11):1695-1705

[3] Lions J-L, Maday Y, Turinici G. A "parareal" in time discretization of PDE's[J]. Comptes Rendus de l'Académie des Sciences - Series I-Mathematics, 2001, 332(7):661-668

[4] Balay S, Buschelman K, Eijkhout V, et al. PETSc Users Manual [EB/OL]. http://www-unix.mcs.anl.gov/petsc/petsc-as/documentation/index.html#Manual

[5] Knepley M. PETSc Tutorial [EB/OL]. http://www-unix.mcs.anl.gov/petsc/petsc-as/documentation/tutorials/index.html

[6] Hovland P D, LMcInnes C. Parallel simulation of compressible flow using automatic differentiation and PETSc[J]. Parallel Computing, 2001, 27(4):503-519

[7] Kelley C T. Iterative Methods for Linear and Nonlinear Equations [M]. Philadelphia: SIAM Press, 1995

[8] Saad Y, Schultz M H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems[J]. SIAM J. Sci. Stat. Comput., 1986, 7:856-869

(上接第108页)

思维整体处理 $\Delta c_2^{NF}[i]$。

(3)对于 $\Delta\Psi_7[i]$，关注 $\Delta\Psi_7[i]$是否等于 $\Delta\Phi_7[i]$，关注进位扩散对其的影响。

(4)在控制差分路径时，关注第32 bit。鉴于第32bit的特殊性，差分[+32]和[-32]在差分分析中认为是相同的。

结束语　本文是基于王小云教授所写的MD5破译的文章，从理论分析和程序验证两方面共同对MD5算法进行研解析。文中以第8步的差分特征为目标，详细解析了F函数的特性、模差分和带符号的异或差分的特点，深入分析了MD5第8步的各bit差分的产生过程，说明了满足其差分特征的条件，对王的文章的进行了部分修正，最后总结了控制差分路径的几个关键点。本文对MD5和其他Hash函数的分析和破译有着重要的作用。

参考文献

[1] Rivest R L. The MD4 message digest algorithm [C]// Advances in Cryptology-Crypto'90. LNCS 537. Springer-Veralg, 1991: 303-311

[2] Rivest R L. The MD5 message-digest algorithm[C]// Request for Comments (RFC 1320). 1992

[3] Zheng Y, Pieprzyk J, Seberry. HAVAL-A one-way hashing algorithm with variable length of output[C]// Advances in Cryptology, Auscrypto'92, LNCS 718. 1992: 83-104

[4] RIPE. Integrity primitives for secure information systems[R]. Final report of RACE integrity primitives evaluation (RIPE-RACE 1040), LNCS 1007, 1995

[5] FIPS 180-1. Secure hash standard[s]. NIST, US Department of Commerce, Springer-Verlag: Washington D C, 1996

[6] FIPS180-2. Secure hash standard[s]. http://csrc.nist.gov/publications

[7] Wang Xiaoyun, Lai Xuejia, Feng Dengguo. Cryptanalysis of the Hash Functions MD4 and RIPEMD[C]// EUROCRYPT 2005, LNCS 3494. 2005: 1-18

[8] Wang Xiaoyun, Yu Hongbo. How to break MD5 and other Hash Functions[C]// EUROCRYPT 2005, LNCS 3494. 2005: 19-35

[9] Wang Xiaoyun, Yu Hongbo. Efficient collision search attacks on SHA-0[C]// Crypto 2005, LNCS 3621. 2005: 1-16

[10] Wang Xiaoyun, Yu Hongbo. Finding collisions in the Full SHA-1 [C]// Crypto 2005, LNCS 3621. 2005: 17-36

[11] 王小云，冯登国，于秀源. 中国科学[J]. 信息科学，2005，35(3)：1-12

[12] Liang Jie, Lai Xuejia. Improved collision attack on Hash Function MD5[J]. Journal of Computer Science & Technology, 2007, 22(1):79-87