文章编号:1001 - 9081(2009)12 - 3174 - 04

破译 MD5 算法关键技术探索

毛 明[1,2,3],秦志光[1],陈少晖[2,3]
(1. 电子科技大学 计算机科学与工程学院,成都 610054;
2. 北京电子科技学院 信息安全系,北京 100070; 3. 西安电子科技大学 通信工程学院,西安 710071)
(chensh@mail.besti.edu.cn)

摘 要:针对 Hash 函数 MD5 算法的结构特点,从明文差分的引入、差分路径的控制和充分条件的确立等方面系统总结了该算法破译过程的关键技术及其主要步骤。首先介绍了破译过程中应用的三种差分的概念,分析了 MD5 算法中非线性函数的性质以及符号差分的扩展、循环左移的特点,然后从整体的分析思想和具体的实践方法两方面对破译 MD5 算法的关键技术进行了探索,以实例详细解析了消息修改技术,对 Hash 函数的破译进行了进一步的研究和探索。

关键词:Hash 函数;MD5;破译;消息修改
中图分类号: TP309 **文献标志码**:A

Exploration of key points for attack of MD5 algorithm

MAO Ming[1,2,3], QIN Zhi-guang[1], CHEN Shao-hui[2,3]
(1. *School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 610054, *China*;
2. *Department of Information Security, Beijing Electronic Science and Technology Institute, Beijing* 100070, *China*;
3. *School of Telecommunication Engineering, Xidian University, Xi'an Shaanxi* 710071, *China*)

Abstract: Based on the structural characteristics of the MD5 algorithm, the authors summarized the key points of deciphering the Hash function MD5: the introduction to the message differential, the control of the differential path and the satisfaction of the sufficient conditions. In the process of deciphering the MD5, three differences and the properties of the non-linear functions were introduced. The extensive form of the signed difference and the affection of the left shift rotation were applied in it. The important technique for attack of the MD5 algorithm, named technique of message modification, was cryptanalyzed in detail with an example. In general, the authors explored the key points of deciphering the MD5 algorithm form both the overall analysis and specific practice.

Key words: Hash function; MD5; attack; message modification

0 引言

Hash 函数是信息安全领域中一个非常重要的基本工具,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,是信息安全领域的研究热点之一。

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。

2004 年 8 月,在美国加州召开的国际密码学年会上,中国研究人员王小云教授等公布了一种破解 MD4、MD5、SHA-0、SHA-1、HAVAL-128 和 RIPEMD 等 Hash 函数的方法[7-11],其攻击算法的复杂度都大大降低,在普通的个人电脑上就可以找到相关碰撞的实例。这一研究成果对现有的 Hash 函数提出了严峻的挑战,促进了新的 Hash 算法的开发研究。

MD5 算法作为 Hash 函数大家庭中的一员,是 MD 结构的典型代表。在相当长的时间内,MD5 算法都广泛应用于信息安全领域,具有其他算法所不可比拟的应用优势。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。

1 预备知识

1.1 MD5 算法描述

MD5 算法是 MD4 算法的改进算法。MD5 算法以 512 位分组来处理输入的信息,且每一分组又被划分为 16 个 32 位子分组。MD5 算法中有 4 个被称之为"链结变量"的 32 位的寄存器,设置好这 4 个寄存器的初始值,就进入了 4 轮的压缩运算。4 轮运算结构相同,但每轮所用的布尔函数、定值参数、循环左移的位数不同。每轮包含 16 步迭代,4 轮共 64 步。经过了 4 轮的压缩处理后,每个链接变量和其相应的初始值相加,最后输出一个 128 位消息摘要。

1.2 模差分、符号差分和异或差分

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、符号差分和模差分等几个重要的概念。

异或差分[8] 设 X 和 X' 是两个 32 位的数,其异或差分为两个数的位的异或运算,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,$X' - X = [5,6,7,8,\cdots,-27]$,这说明 X 和 X' 从第 5 到第 27 位都不同,而且从符号可

收稿日期:2009 - 06 - 30;**修回日期**:2009 - 08 - 24。

作者简介:毛明(1963 -),男,山西稷山人,教授,博士研究生,主要研究方向:信息安全、密码学; 秦志光(1956 -),男,四川隆昌人,教授,博士生导师,主要研究方向:信息安全、网络安全; 陈少晖(1983 -),男,安徽芜湖人,硕士研究生,主要研究方向:信息安全、密码学。

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[8]　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,…,-27],此时用模差分表示为 $X'-X=-2^4$。

1.3　符号差分的扩展

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:

$$\begin{aligned}2^k &= 2^{k+1}-2^k = 2^{k+2}-2^{k+1}-2^k = \cdots = \\ &\quad 2^{n-1}-2^{n-2}-2^{n-3}-\cdots-2^k;0\leqslant k<n\leqslant 32\\ -2^k &= -2^{k+1}+2^k = -2^{k+2}+2^{k+1}+2^k = \cdots = \\ &\quad -2^{n-1}+2^{n-2}+2^{n-3}+\cdots+2^k;0\leqslant k<n\leqslant 32\end{aligned}$$

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:

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

为了便于分析,符号差分可以简化为如下的表示方式:

$$\begin{aligned}[k] &= [k+1,-k] = \\ &\quad [k+2,-(k+1),-k] = \cdots = \\ &\quad [n-1,-(n-2),\cdots-k] = \\ &\quad [32,-31,\cdots,-k] = \\ &\quad [-32,-31,\cdots,-k]\\ [-k] &= [-(k+1),k] = \\ &\quad [-(k+2),(k+1),k] = \cdots = \\ &\quad [-(n-1),(n-2),\cdots,k] = \\ &\quad [-32,31,\cdots,k] = [32,31,\cdots,k]\end{aligned}$$

另外,上面的差分扩展还可以将两个元素联合起来考虑:

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

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。

1.4　符号差分的左循环移位

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:在文献[13]中,设 $2^k=[k+x,-(k+x-1),\cdots,-k]$,其中 $1\leqslant x\leqslant 32-k$,设循环左移的位数为 s(用 $<<<s$ 表示),分3种情况讨论移位后的结果:

当 $k+x+s\leqslant 31$,移位的结果为 $(2^k<<<s)=2^{k+s}$;

当 $k+x+s>31$ 且 $k+s\leqslant 31$,移位的结果为 $(2^k<<<s)=2^{k+s}+1$;

当 $k+s>31$,移位的结果为 $(2^k<<<s)=2^{k+s-32}$。

这三种情况将在具体的差分控制过程中加以应用。特别是第二种情况,必须利用MD5中4个逻辑函数的性质来处理移位后产生的"1"。

同理,可以推算 $-2^k=[-(k+x),(k+x-1),\cdots,k]$ 循环左移 s 位的三种情况。

2　破译 MD5 的关键步骤

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整体思路;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。

2.1　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,这样可以有效地降低计算复杂度;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[8,13-14],都是采用"两个明文块",即1 024位的明文,其中文献[8,13]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[14]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3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。

2.2　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,$b_3=c_3+((b_2+F(c_3,d_3,a_3)+m_{11}+t_{11})<<<22)$,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(在 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,产生后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,必须根据需要来选择;最后,结合第32位的特殊性和第1.3节的内容,要跟踪循环左移对这个差分继承的影响。

2.3　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4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。

从$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Y)$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[7]:

1)当且仅当$Y = Z$,$F(X,Y,Z) = F(\neg X,Y,Z)$。

2)当且仅当$X = 0$,$F(X,Y,Z) = F(X,\neg Y,Z)$。

3)当且仅当$X = 1$,$F(X,Y,Z) = F(X,Y,\neg Z)$。

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 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在有限的范围内。Hash函数中的明文在每一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明文,即在前16步的范围内进行修改,称为单消息修改技术;如果修改的明文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。

3.1 单消息修改技术

在MD5算法的前16步(参照文献[13]中的表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。

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

$a_{3,j} = 0;\quad j = 17,21,23,26,31,32$

$a_{3,k} = b_{2,k};\quad k = 8,13,14,15$

2)附加条件。

$a_{3,10} = 0$

$a_{3,11} = b_{2,11}$

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的修改过程如下:

$$a[3] = (a[3]\hat{}(a[3]\&0xfe7df6e2)\hat{}(b[2]\&0x000074a0)) \mid 0x3c2c8842$$

分成3步来实现:

1)"$a[3]\hat{}(a[3]\&0xfe7df6e2)$",这步将所有充分条件和附件条件所涉及的位都清零。

2)"$b[2]\&0x000074a0$",这步提取了a_3与b_2相关的位。

3)"$\mid 0x3c2c8842$",这步将a_3中必须为1的位置1。

通过以上步骤即可对MD5算法进行相应的单消息修改。

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。

下面以文献[13]的第5.1节为例,分8步来介绍多消息修改技术。

第1~2步 应用第3.1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,这是由于文献[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,这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:$a_5 = b_4 + \Sigma_{16} <<< 22$,$\Sigma_{16} = a_4 + G(b_4,c_4,d_4) + m_1 + 0xf61e2562$。由上一步可知,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:$c_2 = d_2 + \Sigma_6 <<< 17$,$\Sigma_6 = c_1 + F(d_2,a_2,b_1) + m_6 + 0xa8304613$,$c_1$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:$c_5 = d_5 + \Sigma_{18} <<< 14$,$\Sigma_{18} = c_4 + G(d_5, a_5, b_4) + m_{11} + 0x265e5a51$。

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

2)联系第一轮用到m_{11}的地方:$b_3 = c_3 + (b_2 + F(c_3,d_3, a_3) + m_{11} + 0x265e5a51) <<< 22$;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和b_2的变化将分别对下面的4步MD5运算产生影响。在消息修改的第1步已经确定了MD5算法中前16步产生的链接变量的充分条件。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,进而确立前面步骤尚未确定的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第2.3节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 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。

4　结语

本文介绍了 MD5 破译的整体思想和关键技术,并介绍了明文差分的引入原则,作为产生碰撞的第一步,对整个算法的破译起到关键性的作用;分析了 MD5 算法迭代中差分路径的控制方法,利用非线性函数的性质和差分的扩展来引导中间变量差分的继承、产生和消除,最后介绍了单消息修改技术和多消息修改技术在破译 MD5 中的实现方法。

参考文献:

[1] RIVEST R. The MD4 message digest algorithm [C]// CRYPTO '90: Advances in Cryptology. Berlin: Springer-Verlag, 1991: 303 - 311.

[2] RIVEST R L. RFC 1320, the MD5 message-digest algorithm[S]. MIT Laboratory for Computer Science and RSA Data Security, 1992.

[3] ZHENG Y, PIEPRZYK J. HAVAL — A one-way hashing algorithm with variable length of output [C]// AUSCRYPT '92: Advances in Cryptology, LNCS 718. Berlin: Springer-Verlag, 1993: 83 - 104.

[4] PRENEEL B . Integrity primitives for secure information systems [M]. Berlin: Springer-Verlag, 1995: 116 - 125.

[5] FIPS 180-1, secure Hash standard[S]. NIST, US Department of Commerce, Federal Information Processing Standard PUB, 1996: 1 - 20.

[6] FIPS 180-2, secure Hash standard[S]. NIST, US Department of Commerce, Federal Information Processing Standard PUB, 1997: 1 - 10.

[7] WANG X Y, LAI X J, FENG D G. Cryptanalysis of the Hash functions MD4 and RIPEMD[C]// EUROCRYPT 2005, LNCS 3494. Berlin: Springer-Verlag, 2005: 1 - 18.

[8] WANG X Y, YU H B. How to break MD5 and other Hash functions [C]// EUROCRYPT 2005, LNCS 3494. Berlin: Springer-Verlag, 2005: 19 - 35.

[9] WANG X Y, YU H B. Efficient collision search attacks on SHA-0 [C]// CRYPTO 2005, LNCS 3621. Berlin: Springer-Verlag, 2005: 1 - 16.

[10] WANG X Y, YU H B, Finding collisions in the Full SHA-1[C]// CRYPTO 2005, LNCS 3621. Berlin: Springer-Verlag, 2005: 17 - 36.

[11] 王小云,冯登国,于秀源. HAVAL-128 的碰撞攻击[J]. 中国科学: E 辑,2005, 35(3): 1 - 12.

[12] LIANG J, LAI X J. Improved collision attack on Hash function MD5 [J]. Journal of Computer Science & Technology, 2007, 22(1): 79 - 87.

[13] XIE T, FENG D G. A new differential for MD5 with its full differential path [EB/OL]. [2009 - 04 - 10]. http://eprint.iacr.org/2008/230.pdf.

[14] XIE T, LIU F B. Could 1-MSB input differential be the fastest collision attack for MD5 [EB/OL]. [2009 - 04 - 20]. http://eprint.iacr.org/2008/391.

(上接第 3173 页)

[13] BECCHI M, CROWLEY P. An improved algorithm to accelerate regular expression evaluation [C]// Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems. New York: ACM Press, 2007, 145 - 154.

[14] FICARA D, GIORDANO S, PROCISSI G, *et al.* An improved DFA for fast regular expression matching [J]. ACM SIGCOMM Computer Communication Review, 2008, 38(5): 29 - 40.

[15] SMITH R, ESTAN C, JHA S. Xfa: Faster signature matching with extended automata [C]// Proceedings of the 2008 IEEE Symposium on Security and Privacy. Washington, DC: IEEE, 2008: 187 - 201.

[16] SMITH R, ESTAN C, JHA S, *et al.* Deflating the big bang: Fast and scalable deep packet inspection with extended finite automata [C]// Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication. New York: ACM Press, 2008: 207 - 218.

[17] BECCHI M , CADAMBI S . Memory - efficient regular expression search using state merging [C]// INFOCOM 2007: 26th IEEE International Conference on Computer Communications. Washington, DC: IEEE, 2007: 1064 - 1072.

[18] KUMAR S, VARGHESE G. Curing regular expressions matching algorithms from insomnia, amnesia and acalculia [C]// Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems. Washington, DC: IEEE, 2007: 155 - 164.

[19] BECCHI M, CROWLEY P. Extending finite automata to efficiently match Perl-compatible regular expressions [C]// Proceedings of the 2008 ACM CoNEXT Conference. New York: ACM Press, 2008.

[20] YU F, CHEN Z, DIAO Y, *et al.* Fast and memory-efficient regular expression matching for deep packet inspection [C]// Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems. New York: ACM Press, 2006: 93 - 102.

[21] BECCHI M, CROWLEY P. A hybrid finite automaton for practical deep packet inspection [C]// Proceedings of the 2007 ACM CoNEXT Conference. New York: ACM Press, 2007: 1 - 12.

[22] KONG S, SMITH R, ESTAN C. Efficient signature matching with multiple alphabet compression tables [C]// Proceedings of the 4th international Conference on Security and Privacy In Communication Networks. New York: ACM Press, 2008.

[23] Perl compatible regular expressions [EB/OL]. [2009 - 04 - 11]. http://www.pcre.org/ .

[24] 郝克刚,段振华,李新. 论回溯自动机[J]. 计算机学报, 1990, 13(5): 340 - 348.

[25] HOPCROFT J E, MOTWANI R, ULLMAN J D. Introduce to automata theory, languages, and computation [M]. 3rd ed. Upper Saddle River, New Jersey, USA: Addison-Wesley Professional, 2007.

[26] BECCHI M, CROWLEY P. Efficient regular expression evaluation: Theory to practice [C]// Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems. New York: ACM Press, 2008: 50 - 59.

[27] BECCHI M, FRANKLIN M, CROWLEY P. A workload for evaluating deep packet inspection architectures [C]// IISWC 2008: IEEE International Symposium on Workload Characterization. Washington, DC: IEEE, 2008: 79 - 89.

[28] 陈曙晖,苏金树,范慧萍,等. 一种基于深度报文检测的 FSM 状态表压缩技术[J]. 计算机研究与发展, 2008, 45(8): 1299 - 1306.