# 带认证邮局协议的密钥恢复攻击

刘凡保[1) ] 谢 涛[1)] 冯登国[2)]
[1)](国防科学技术大学计算机学院 长沙 410073)
[2)](中国科学院信息安全国家重点实验室 北京 100190)

**摘 要** 作者提出了一种新的针对带认证邮局协议的密钥恢复攻击,能够更快地恢复出密钥并能够恢复更多的密钥字符.基于通道技术和高级消息修改技术,提出了一种"群满足方案"来确定性地满足分而治之策略下最后一个通道首三步的所有充分条件,籍此提高MD5(Message Digest Algorithm 5)碰撞对搜索的效率.并提出了一些新的通道来控制MD5碰撞对消息的更多比特的取值,比如可以构造出352比特值确定的MD5碰撞对.通过这些技术改进了多位信息确定的MD5碰撞对搜索效率,应用到APOP的密钥恢复攻击中不仅能够快速恢复长达31个字符的密钥,而且能够在实际时间内恢复长达43个字符的密钥.

**关键词** 带认证邮局协议;挑战和响应;密钥恢复;通道;群满足方案
**中图法分类号** TP309 **DOI号**: 10.3724/SP.J.1016.2012.01927

## Password Recovery Attack to Authentication Post Office Protocol

LIU Fan-Bao[1)] XIE Tao[1)] FENG Deng-Guo[2)]
[1)](*School of Computer, National University of Defense Technology, Changsha* 410073)
[2)](*State Key Laboratory of Information Security, Chinese Academy of Sciences, Beijing* 100190)

**Abstract** In this paper, we propose a new password recovery attack to Authentication Post Office Protocol (APOP), which can recover more password characters and faster. First, based on tunnel and advanced message modification technologies, we propose a "Group Satisfaction Scheme"to satisfy determinately all conditions of the first three successive steps of the last tunnel, to further improve Message Digest Algorithm 5 (MD5) collision searching efficiency. Second, we propose some new tunnels to generate more meaningful characters during MD5 collision searching; for example, we can construct an MD5 collision pair with as many as 352 fixed bits. Combining with these technologies, we can improve the efficiency of MD5 collision searching with high number of chosen bits, hence, we can recover APOP passwords with 31 characters extremely fast, and can also recover passwords as long as 43 characters in practical time.

**Keywords** authentication post office protocol; challenge and response; password recovery; tunnel; group satisfaction scheme

## 1 引 言

带认证邮局协议(Authentication Post Office Protocol——Version 3,APOP)[1]被广泛应用于邮局协议——版本3(POP3)以提供用户原始身份验证和抗重放攻击.APOP是基于散列函数MD5的挑战和响应的密钥认证协议.APOP的身份认证包

收稿日期:2012-05-18;最终修改稿收到日期:2012-07-11.本课题得到国家核高基重大专项课题(2010ZX01037-001-001)、国家"九七三"重点基础研究发展规划项目基金(2007CB311202)和国家自然科学基金(61070228)资助.**刘凡保**,男,1982年生,博士研究生,主要研究方向为密码分析、网络安全和可信计算.E-mail:liufanbao@gmail.com.**谢 涛**,男,1966年生,博士,研究员,主要研究领域为演化计算、密码学和信息安全.**冯登国**,男,1965年生,博士,研究员,主要研究领域为网络与信息系统安全、密码理论与技术.

括两方参与者,即认证服务器(认证者)和用户(被认证者).持有和存储电子邮件的认证服务器向要访问其邮件的用户发送一个满足一定限制且随机和唯一的挑战数据 $C$.之后,用户使用散列函数 MD5 对收到的挑战 $C$ 和持有的共享密钥 $P$ 进行散列计算,并将计算的散列值 $R$ 发送至认证服务器.最后,基于共享密钥 $P$,认证服务器重做与用户一致的散列计算并将结果 $R'$ 与接收到的散列值 $R$ 比较,若二值相同,则认证成功.否则,认证失败,中止连接.

### 1.1 以往的工作

1996 年,基于散列碰撞的分而治之策略,Preneel 和 van Oorschot[2] 描述了如何恢复基于秘密信封方案 MAC 的密钥.他们使用生日攻击方法获得碰撞对,对于广泛使用的散列函数如 MD5[3] 和 SHA-1[4] 等,该方法在现有条件下是计算不可行的.然而,当一些流行散列函数的弱碰撞安全被攻击和破坏[5-8] 且相应碰撞对搜索的效率也极大提高后[9],一系列针对 APOP 协议的密钥恢复攻击被陆续提出[10-13],且所有的攻击都是基于选择挑战的.

在我们的工作之前,已知 APOP 攻击的最好结果[12] 是利用 den Boer 和 Bosselaers[14] 提出的一种伪碰撞——dBB 碰撞,利用该碰撞可以得到更多的自由消息位和恢复出更多的密钥字.例如,用来恢复 APOP 所使用密钥的前 11 个字符的 MD5 碰撞对可以在不到 1 s 的时间内生成,用来恢复 APOP 所使用密钥的前 31 个字符的 MD5 碰撞对也可以在实际时间内生成[12].然而,这种方法的缺陷在于:为了利用 dBB 碰撞必须以 $2^{42}$ 次 MD5 散列计算的复杂度首先生成一个额外的"IV 桥".在文献[13]中,提出了"比特自由碰撞对"的概念,即一对部分比特值固定的消息总会碰撞而无论其未固定比特如何取值,利用"比特自由碰撞对"可以降低 APOP 攻击所需的挑战数目.用来恢复 APOP 所使用密钥的前 11 个字符的 1 比特自由碰撞对可以经 $2^{46}$ 次 MD5 散列后生成,可以减少所需发送挑战数的一半.

### 1.2 我们的工作

我们提出两个新的通道 $q_1'$ 和 $q_1$ 分别生成具有 320 位和 352 位信息确定的 MD5 碰撞对.

基于通道技术(tunnel)和高级消息修改技术,我们提出一种群满足方案——同时确定性地满足最后一个通道首三步的所有充分条件,进一步改进了 MD5 碰撞对搜索的效率.

结合分而治之和群满足方案降低具有多位信息确定的 MD5 碰撞对搜索的复杂度.在普通的个人 PC 机上,生成恢复 APOP 所使用密钥的前 11 个字符的 MD5 碰撞对的平均时间为 0.08 s,生成恢复 APOP 所使用密钥的前 31 个字符的 MD5 碰撞对的平均时间为 0.15 s,前 39 个字符的碰撞对的所需平均时间为 4.13 s.此外,应用这些技术还可以降低搜索 1 比特自由碰撞对的复杂度.

另外我们提出一种在本地 PC 上恢复 APOP 所使用密钥的快速密钥恢复攻击.我们能够在不到 1 min的时间内恢复出长达 11 个字符的密钥内容,在 4 min 左右能够恢复出长达 31 个字符的密钥内容,而长达 43 个字符的密钥也能够在实际可达时间内破解.该攻击模拟了一种可能由恶意软件发起的攻击,进一步论证了 APOP 的安全性已经完全破坏,因而基于 MD5 的网络应用程序的安全性需要进一步地评估[15-16].

本文第 2 节简要介绍 APOP 协议及相关概念;第 3 节对一些相关工作进行比较;第 4 节详细介绍具有多位信息确定的 MD5 碰撞对搜索;第 5 节模拟一个 APOP 密钥恢复攻击;在第 6 节对本文进行总结.

## 2 基础知识

### 2.1 MD5 算法描述

MD5[3] 是一种典型的基于 MD 结构(Merkle-Damgård Structure)的迭代散列函数.其接收可变长度的消息 $M$(输入消息 $M$ 的长度一般为 $1\leq|M|\leq 2^{64}-1$ bit)作为输入且输出固定长度为 128 bit 的散列值 $MD5(M)$.

#### 2.1.1 预处理

在对输入消息 $M$ 进行散列前需要对其进行预处理,分为如下 3 个阶段:

(1) 消息 $M$ 首先被填充至分组块 512 bit 的倍数,其中 $M$ 附加一个 1 和若干个 0 至 448 mod 512,最后附加 64 bit 的 $M$ 的长度信息.

(2) 填充后的消息 $M'$ 被分成若干个 512 bit 的分组块($M_0, M_1, \cdots, M_{(|M'|/512-1)}$).

(3) 每个 512 位的分组块 $M_i$ 被分割为 16 个 32 比特字($m_0, m_1, \cdots, m_{15}$).

#### 2.1.2 MD5 压缩函数

每个分组块依次被 MD5 的压缩函数 $CF$ 处理,压缩函数 $CF$ 接收 512 bit 的 $M_i$ 和 128 bit 的链接变量 $H_i$ 作为其输入并输出 128 bit 的 $H_{i+1}$.初始链接变量 $H_0$ 设置为固定常数 $a_0=0$x67452301, $b_0=$

0xefcdab89，$c_0$=0x98badcfe，$d_0$=0x10325476. MD5 算法的迭代过程如下所示，其中 $H_n$ 为散列值 $MD5(M)$.

$$H_1=CF(M_0,H_0),\cdots,H_n=CF(M_{n-1},H_{n-1}).$$

压缩函数 $CF$ 由 64 步构成，其中 1～16 步、17～32 步、33～48 步以及 49～64 步分别称为第 1 轮 $r_1$、第 2 轮 $r_2$、第 3 轮 $r_3$ 和第 4 轮 $r_4$. 设 $q_i(1\leq i\leq 64)$ 表示压缩函数第 $i$ 步的 32 bit 状态字，而 $q_{i,j}$ 表示状态字 $q_i$ 的第 $j$ 比特值($0\leq j\leq 31$)，并将链接变量初始化为 $q_{-3}=a_0$，$q_0=b_0$，$q_{-1}=c_0$，$q_{-2}=d_0$，则状态字 $q_i(1\leq i\leq 64)$ 的更新如式(1)所示：

$$q_i=q_{i-1}+(q_{i-4}+f_i(q_{i-1},q_{i-2},q_{i-3})+w_i+t_i)\lll s_i \tag{1}$$

状态字 $q_i$ 的更新使用模 $2^{32}$ 加法＋、循环左移操作≪、轮函数 $f_i$、消息字 $w_i$ 和常数 $t_i$ 进行.

轮函数 $f_i$ 的具体信息如式(2)所示：

$$f_i=\begin{cases}F(B,C,D)=(B\wedge C)\vee(\neg B\wedge D), & i\in r_1\\ G(B,C,D)=(B\wedge D)\vee(C\wedge\neg D), & i\in r_2\\ H(B,C,D)=B\oplus C\oplus D, & i\in r_3\\ I(B,C,D)=C\oplus(B\vee\neg D), & i\in r_4\end{cases} \tag{2}$$

其中，⊕、∧、∨ 与 ¬ 分别表示逐位“异或”操作、“与”操作、“或”操作和“非”操作，且 $B,C,D$ 分别表示32 比特字. 本文中若未作特别说明，则 $f_i$ 表示 $f_i=(q_{i-1},q_{i-2},q_{i-3})$.

而消息字 $w_i$ 为消息字 $(m_0,m_1,\cdots,m_{15})$ 之一，其取值称为 MD5 的消息字扩展，细节如式(3)所示：

$$w_i=\begin{cases}m_{i-1}, & i\in r_1\\ m_{(5i-4)\bmod 16}, & i\in r_2\\ m_{(3i+2)\bmod 16}, & i\in r_3\\ m_{7(i-1)\bmod 16}, & i\in r_4\end{cases} \tag{3}$$

常数 $t_i$ 的取值定义为如式(4)所示：

$$t_i=\lfloor 2^{32}\times|\sin(i)|\rfloor \tag{4}$$

$\lll s_i$ 表示循环左移 $s_i$ 比特，相应的 $\ggg$ 表示循环右移. 状态字更新的每步循环移位操作细节如式(5)所示：

$$(s_i,s_{i+1},s_{i+2},s_{i+3})=\begin{cases}(7,12,17,22), & i=1,5,9,13\\ (5,9,14,20), & i=17,21,25,29\\ (4,11,16,23), & i=33,37,41,45\\ (6,10,15,21), & i=49,53,57,61\end{cases} \tag{5}$$

若 64 步状态更新完成则完成了 MD5 压缩函数的一次调用，需要利用最后 4 个状态字对链接变量 $H_i$ 进行更新.

### 2.2　MD5 碰撞对性质

若存在两个任意不同的消息 $m$ 和 $m'(m,m'\in\{0,1\}^*)$ 满足 $MD5(m)=MD5(m')$，则 $m$ 和 $m'$ 称作一碰撞对. 设 $|M|$ 为消息 $m$ 的长度并为分组块 512 bit 的倍数，若进一步满足 $|m|=|m'|$，则对任意消息 $x$，满足

$$MD5(m\parallel x)=MD5(m'\parallel x) \tag{6}$$

式(6)为 APOP 密钥恢复攻击的基础.

### 2.3　消息修改技术

为了有效地搜索碰撞对，需要利用一些消息修改技术满足维持 MD5 差分路径的大量充分条件. 比如在王小云等人[5]提出的碰撞攻击中，第 1 个分组块的差分路径有超过 200 个条件，但是优化后的计算复杂度只需 $2^{24.8}$ 次 MD5 散列计算[17].

文献[5]中提出了基本消息修改和高级消息修改技术满足维持差分路径所需的充分条件. 根据碰撞攻击的属性，攻击者可以完全控制消息字 $(m_0,m_1,\cdots,m_{15})$ 的取值，因此，基本消息修改技术可以很容易在第一轮 $r_1$ 实现. 若第 1 轮状态字 $q_i$ 的某比特值不满足条件，则攻击者只需修改相应的 $m_{i-1}$ 的值即可. 事实上，完全可以直接设置 $q_i(1\leq i\leq 16)$ 的充分条件为真，然后再计算对应 $w_i=m_{i-1}$ 的值，如式(7)所示：

$$m_{i-1}=w_i=(q_i-q_{i-1})\ggg s_i-q_{i-4}-f_i-t_i,\ i\in r_1 \tag{7}$$

为了满足第 2 轮 $r_2$ 的一些充分条件，可以通过修改第 1 轮 $r_1$ 的对应消息字 $m_i$，并维持因消息字修改所导致的第 1 轮差分路径的变化不传入第 2 轮(差分路径不变)，即为高级消息修改技术. 举例说，如果充分条件 $q_{25,31}=q_{0,31}$ 不满足，则我们可以通过修改 $w_{25,26}=m_{9,26}$ 的值或者说在未进行循环左移前引进一个 $2^{26}$ 的差分，状态字 $q_{25}$ 的更新如式(8)所示：

$$q_{25}=q_{24}+(q_{21}+f_{25}+(\underline{w_{25}=m_9})+t_{25})\lll 5 \tag{8}$$

由于第 1 轮 $r_1$ 中状态字 $q_{10}\sim q_{14}$ 的值直接受到 $m_9$ 的影响，若我们直接修改 $m_9$ 的相关值，则它们的对应取值都将改变，从而引起状态字 $q_{10}\sim q_{14}$ 的改变(如式(10)～式(14)所示)，即状态字 $q_{25}$ 之前的差分路径已经破坏. 因此直接的消息修改毫无意义，即在此情形下基本消息修改技术不能使用.

$$q_9=q_8+(q_5+f_9+(w_9=m_8)+t_9)\lll 7 \tag{9}$$

$$\underline{q_{10}}=q_9+(q_6+f_{10}+(w_{10}=\underline{m_9})+t_{10})\lll 12 \tag{10}$$

$$q_{11}=\underline{q_{10}}+(q_7+f_{11}+(w_{11}=\underline{m_{10}})+t_{11})\lll 17 \tag{11}$$

$$q_{12}=q_{11}+(q_8+f_{12}+(w_{12}=\underline{m_{11}})+t_{12})\lll 22 \tag{12}$$

$$q_{13}=q_{12}+(q_9+f_{13}+(w_{13}=\underline{m_{12}})+t_{13})\lll 7 \tag{13}$$

$$q_{14}=q_{13}+(\underline{q_{10}}+f_{14}+(w_{14}=\underline{m_{13}})+t_{14})\lll 12 \tag{14}$$

为了克服这个缺点，可选项之一是保持状态字 $q_{11}\sim q_{14}$ 的值不变，而修改相应的消息字 $m_{10}\sim m_{13}$ 的值，如式(10)～式(14)所示. 然而，问题并未能真正解决. 我们注意到 $m_{10}$ 和 $m_{11}$ 的值已经分别在第 2 轮 $r_2$ 更新状态字 $q_{22}$ 和 $q_{19}$ 时使用，也即之前所作修改消息字 $m_{10}$ 或者 $m_{11}$ 的值以维持 $r_1$ 的差分路径将导致第 2 轮 $q_{25}$ 之前的差分路径改变，因此该方法不可行.

为了另寻出路，我们注意到消息字 $m_8$ 将在第 2 轮 $q_{25}$ 之后的 $q_{28}$ 中使用($w_{28}=m_8$)，即可以通过修改状态字 $q_9$ 的值但同时保持状态字 $q_{10}$ 的值不变以达到间接修改 $m_9$ 的目的. 在状态字更新式 $q_{10}$ 中翻转 $m_{9,26}$ 的值，则循环左移后将在 $q_{10}$ 引入一个差分 $2^6$. 所以在式(10)中翻转 $q_{9,6}$ 的值，并设置一个额外的充分条件 $q_{8,6}=q_{7,6}$ 以保证 $f_{10}=f_{10}(q_{9,6},q_{8,6},q_{7,6})$ 的值不变. 然后从值不变的 $q_{10}$ 中重新计算 $m_9$，如式(15)所示，则获得了期望的差分 $2^{26}$.

$$\underline{m_9}=(q_{10}-\underline{q_9})\ggg 12-q_6-f_{10}-t_{10} \tag{15}$$

进一步，需要设置额外的充分条件 $q_{10,6}=0$ 来保证在更新状态字 $q_{11}$ 时 $m_{10}$ 和 $q_{11}$ 的值不变. 类似设置条件 $q_{11,6}=1$ 以保证在更新状态字 $q_{12}$ 时 $m_{11}$ 和 $q_{12}$ 的值不变. 最后，重新计算 $m_{12}$ 的值以保证 $q_{13}$ 的值也不变，保证了修改 $m_8$ 的值对第 1 轮 $q_{10}$ 之后的差分路径没有影响.

如是，通过高级消息修改技术以概率 1 满足了充分条件 $q_{25,31}=q_{0,31}$，并且维持了 $q_{25}$ 之前的差分路径不变.

### 2.4 通 道

通道的概念由 Klima[9] 提出，其本质优势就是基于自由消息位不仅实现了穷举搜索，而且最重要的是将搜索的起始点后移，从而极大降低了碰撞搜索的复杂度.

dBB 碰撞有 47 个充分条件需要满足($q_{i,31}=q_{0,31}\{i\neq 64, i\in(r_1,r_2,r_4)\}$). 在 dBB 碰撞搜索中，状态字 $q_{10}$ 有充分条件 $q_{10,31}=q_{0,31}$ 需要满足，在状态字 $q_9$ 中有充分条件 $q_{9,31}=q_{0,31}$ 需要满足，因此，$q_{9,31}$ 的值一旦确定满足则不能再更改，否则将会破坏之前的差分路径. 除了保留 $q_{9,6}$ 做高级消息修改以满足条件 $q_{25,31}=q_{0,31}$，状态字 $q_9$ 仍有 30 bit 的信息无需特别设定，即为自由位. 在状态字 $q_{10}$ 设置额外条件 $q_{10}=2^{31}\times(1\wedge q_{0,31})+0\text{x}00000000$；在 $q_{11}$ 中设置条件 $q_{11}=2^{31}\times(1\wedge q_{0,31})+0\text{x}7\text{fffffff}$，则在第 25 步($q_{25}$)中，可以通过翻转 $q_9$ 的任意自由位来启动一个新的碰撞对搜索而无需担心影响 $q_{25}$ 之前的差分路径. 在状态字 $q_{25}$ 之后，dBB 的差分路径有 22 个充分条件需要随机满足，而 $q_9$ 通道有多达 30 个自由位，也即在 $q_9$ 通道之内确定能够找到多个 dBB 碰撞对. 因此，一旦到达 $q_9$ 通道，dBB 碰撞对搜索的起始点就是 $q_{25}$，而不是未应用通道技术前的 $q_1$，从而极大提高了碰撞对搜索效率.

应用 $q_9$ 通道之后，根据消息修改技术的原理，变化的 $q_9$ 对 MD5 第一轮差分路径所产生的影响在 $q_{13}$ 之后完全消除. 即 $q_9$ 通道的应用不会影响到 $m_{13}$、$m_{14}$ 和 $m_{15}$ 的取值. 因此，可以固定 $m_{13}$、$m_{14}$ 和 $m_{15}$ 的值(96 位信息确定)来产生 MD5 碰撞，即对 APOP 密钥的可能字进行猜测(前 11 个字符).

### 2.5 APOP 协议

带认证邮局协议(APOP)是一种流行的基于散列函数的挑战和响应认证协议，一般被邮件服务器用来认证将要访问存储在服务器端邮件的用户身份. APOP 是邮局协议——版本 3(POP3)[1] 的可选安全命令，用以提供额外的安全性，以避免在不安全的网络上传输明文密钥. 设用户和服务器之间预先共享的密钥为 *passwd*，由于邮件访问总是由用户发起，APOP 的认证过程如下：

(1) 用户向认证服务器发送一个连接请求.

(2) 认证服务器随机产生一个满足一定限制条件的挑战数据 $C$(格式化为一个消息标识符)，将 $C$ 发送至用户.

(3) 用户将其共享密钥 *passwd* 附加于接收到的挑战数据 $C$ 之后，如式(16)计算其响应值 $R$，然后将 $R$ 发送至认证服务器.

$$R=MD5(C\parallel passwd) \tag{16}$$

(4) 一旦接收到响应 $R$，认证服务器使用共享密钥如式(16)计算 $R_S$. 服务器校验 $R_S=R$ 是否成立，若是，则用户通过验证. 否则，验证失败.

APOP 协议的优势在于：若散列函数 MD5 为单向散列函数，应用挑战和响应的认证协议 APOP 后，网络监听者在一个 POP3 会话中将不会得到任何有关明文密钥的信息. APOP 中挑战数据 $C$ 为随机产生，该策略可以有效防止针对 POP3 会话的重

放攻击.

APOP 挑战数据的限制条件为

(1) 挑战数据必须以字符'<'(十六进制数 0x3c)开头,且包含字符'@'(0x40).

(2) 挑战数据必须以字符'>'(0x3e)结尾.

(3) 挑战数据中间不能够包含任何'NULL'(0x0)、'<'、'>'和'\n'(0x0a)字符.

## 3 相关工作

### 3.1 密钥恢复攻击

用户首先使用其邮箱帐号和密钥登录邮件服务器,然后才能访问其邮件. 由于帐号一般都是以明文传送,为了保证其邮箱帐号的安全性,密钥的保密性成了唯一的选择,APOP 的设计正是为了不在非信任的网络中传输明文密钥. 然而,仍然有其他的方法来攻击用户的密钥,比如社会工程、特洛伊木马、穷举密钥搜索攻击和字典攻击等.

3.1.1 穷举密钥攻击

穷举密钥攻击不仅直接而且简单,但是不太容易完成. 比如一个极端的例子,若一个密钥仅包含一个二进制位,则其要么为'0'要么为'1'. 我们仅需要一次试探,而不管该试探是否正确我们都能立刻知道有关密钥的正确答案. 事实上,一般的密钥由 52 个字母和 10 个数字构成(简单起见,我们忽略了特殊字符),如果该密钥仅有一个字符,应用穷举密钥攻击在最坏的情况下需要 61 次试探就能得到正确的密钥. 但是,当密钥长度为 8 个字符时,应用穷举攻击则需要大概 $62^8-1\approx 2^{47}$ 次试探才能获得答案,该计算量对于一般的计算机来说不太可行. 因此,一般认为 8 位以上密钥的安全性较高.

3.1.2 "切片"密钥恢复攻击

若 MD5 碰撞对$(m,m')$中的两个消息长度相等且为分组块 512bit 的倍数,则对任意消息 $x$,总是满足 $MD5(m\parallel x)=MD5(m'\parallel x)$. Preneel 和 van Oorschot[2]应用该属性实现分而治之策略"切片"攻击秘密信封方案 MAC 中使用的密钥. 其攻击原理为:首先攻击者猜测首个未知的密钥比特的内容为 $b$,然后攻击者据此生成一个最后一比特值为 $b$ 的碰撞对$(m,m')$;其次,攻击者从消息 $m$ 和 $m'$ 中移除猜测值 $b$,并将不含猜测值的两个消息进行在线 MAC 查询;最后,比较收到的两个响应值,若两值相等,则猜测比特值 $b$ 正确. 否则,猜测错误. 如果能够对长度为 64 bit 的密钥进行分而治之的密钥恢复攻击,则只需要 64 次查询(每次查询都必须首先生成一个碰撞对消息),而不是穷举搜索所需的 $2^{64}-1$"次"查询.

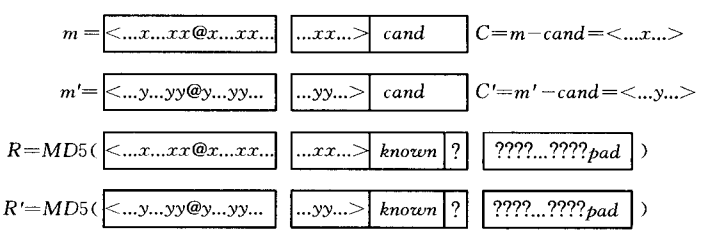



图 1　APOP"切片"密钥恢复攻击

在 APOP"切片"密钥恢复攻击的情形中,由于挑战数据必须为字节的倍数,因此,攻击者应用分而治之策略来进行 APOP 攻击时,密钥恢复的基本单位是一个字节而不是一个比特. APOP 逐字节密钥恢复攻击的总体步骤如下(攻击者伪装成认证服务器):

(1) 生成一个末尾包含有可能密钥字 *cand* 的碰撞对$(m,m')$,其中 *cand* 包含已经恢复的初始化为'NULL'的若干密钥字 *known* 部分和正在猜测的字符 $l$. 从碰撞对消息中分离出挑战数据 $C=m-cand$ 和 $C'=m'-cand$,先将挑战 $C$ 发送至用户.

(2) 从用户处接收响应 $R$,然后将另一挑战 $C'$ 发送至用户.

(3) 从用户处接收另一响应 $R'$,并比较 $R=R'$ 是否成立. 若成立,则表明此次猜测的字符 $l$ 是正确的,将 $l$ 添加至已恢复密钥字 *known* 的尾部,继续下一个字符 $l'$ 的猜测;否则,猜测错误,转至步 1 以进行字符 $l$ 的下一个猜测.

### 3.2 已有的 APOP"切片"密钥恢复攻击

2007 年,Leurent[10]和 Sasaki 等人[11]分别独立提出了针对 APOP 的密钥恢复攻击,该攻击只能够恢复出 APOP 密钥中的前 3 个字符. 该攻击的局限性在于其使用王小云等人[5]提出的 MD5 差分路径,由于该差分攻击的输入差分 $m_{14}$ 的最高位(MSB)存在差分,而 MD5 使用小头(little-endian)的数据存储方式,导致在 APOP 密钥恢复攻击中仅有 $m_{15}$ 可用作密钥字符的猜测(挑战数据 $C$ 必须以字符'>'结尾,因此,仅有 3 个字符可以被使用).

2008 年，Sasaki 等人[12]对 APOP 的密钥恢复攻击进行了改进，通过生成一个“IV 桥”来使用 dBB 碰撞，利用 dBB 碰撞的具有更多自由消息位来恢复更多的密钥字. 改进后，用来恢复 APOP 密钥前 11 个字符的 MD5 碰撞对能够在约 1 s 的时间内找到. 该攻击方法的局限性在于：使用 dBB 碰撞进行密钥恢复攻击前，必须以大约 $2^{42}$ 次 MD5 散列计算构造一个额外的“IV 桥”.

2009 年，通过提出“比特自由碰撞”的概念，Wang 等人[13]进一步对 APOP 的密钥恢复攻击进行了改进，主要在于能够有效减少需发送挑战的数目.“1 比特自由碰撞”意味着若产生原始 MD5 碰撞对所需的复杂度为 $2^t$ 次 MD5 散列计算，则 1 比特自由碰撞的计算复杂度为 $2^{2t}$ 次 MD5 散列计算. 他们通过产生“1 比特自由碰撞”将恢复 APOP 密钥前 11 个字符所需挑战数目减少了一半.

## 4 多位信息确定的碰撞对搜索

为了在 APOP 密钥恢复攻击中应用分而治之策略，首要的问题是有效解决多位信息确定的碰撞对搜索. 本节中，我们提出两个新的通道来控制碰撞对的更多比特值从而恢复更多的密钥字，而且提出一个群满足方案来改进碰撞搜索的效率；结合群满足方案和分而治之策略，实现高效的基于两个分组块且具有多位信息确定的碰撞对搜索，总体策略如下：

(1) 将第一个输入分组块的消息差分设置为 $\Delta M_0=(\Delta m_8=2^{31})$，该差分在文献[18]中提出，并将该分组块当作“IV 桥”. 具有该消息差分的两分组块被 MD5 散列函数处理后，输入的初始链接变量差分 $\Delta H_0=0$ 转变成输出链接变量差分 $\Delta H_1=\pm(2^{31},2^{31},2^{31},2^{31})$，而 $\Delta H_1$ 正是 dBB 碰撞所需要的输入链接变量差分(dBB 条件差分).

(2) 将第 2 个分组块的输入差分设置为 $\Delta M_1=0$ 以使用 dBB 碰撞，在该阶段运用新的通道、群满足方案和分而治之策略以控制更多消息位的取值.

### 4.1 新的通道

为了能够恢复出 APOP 密钥的更多字符，我们在 $q_{20}$ 步和 $q_{17}$ 步分别提出了新的 $q_1'$ 和 $q_1$ 通道.

#### 4.1.1 $q_{20}$ 的 $q_1'$ 通道

$q_{20}$ 应用消息字 $m_0$ 进行状态字更新，注意到状态字 $q_2$ 有充分条件 $q_{2,31}=q_{0,31}$ 需要满足，状态字 $q_1$ 有充分条件 $q_{1,31}=q_{0,31}$ 需要满足，因此，$q_{1,31}$ 的值一旦确定满足则不能再更改，否则将会破坏之前的差分路径. 除了保留 $q_{1,18}$ 做高级消息修改以满足条件 $q_{20,31}=q_{0,31}$，状态字 $q_1$ 仍有 30 bit 的信息无需特别设定，也即为自由位.

由于消息字 $m_1$ 已经在第 2 轮 $q_{20}$ 之前的 $q_{17}$ 使用，所以不能再对其进行任何修改，否则将破坏 $q_{20}$ 之前的差分路径. 但是注意到 $m_2\sim m_5$ 都未在 MD5 的第 2 轮的 $q_{20}$ 之前出现，因此，翻转 $q_1$ 后，根据 $m_1$ 重新计算 $q_2$，再直接修改相应的 $m_2\sim m_5$ 来保持第 1 轮的 $q_3\sim q_6$ 差分路径. 通过以上方式可以成功构造 $q_{20}$ 的 $q_1'$ 通道.

在第 20 步($q_{20}$)中，我们可以通过翻转 $q_1$ 的任意自由位来启动一个新的碰撞对搜索而无需担心影响 $q_{20}$ 之前的差分路径. 而在状态字 $q_{20}$ 之后，dBB 的差分路径有 27 个充分条件需要随机满足，而 $q_1'$ 通道有多达 30 个自由位，也就是说，在 $q_1'$ 通道之内确定能够找到多个 dBB 碰撞对，因此，dBB 碰撞对搜索的起始点就是 $q_{20}$.

应用 $q_1'$ 通道之后，根据消息修改技术的原理，$q_2$ 对 MD5 第一轮差分路径所产生的影响在 $q_6$ 之后完全消除，即 $q_1'$ 通道的应用，不会影响到 $m_6\sim m_{15}$ 的取值. 因此，可以固定 $m_6\sim m_{15}$ 的值(320 bit，40 个字符)来对 APOP 的可能密钥字进行猜测.

#### 4.1.2 $q_{17}$ 的 $q_1$ 通道

$q_{17}$ 应用消息字 $m_1$ 来进行状态字更新，注意到状态字 $q_2$ 有充分条件 $q_{2,31}=q_{0,31}$ 需要满足，状态字 $q_1$ 有充分条件 $q_{1,31}=q_{0,31}$ 需要满足，因此，$q_{1,31}$ 的值一旦确定满足则不能再更改，否则将会破坏之前的差分路径. 除了保留 $q_{1,6}$ 做高级消息修改以满足条件 $q_{17,31}=q_{0,31}$，状态字 $q_1$ 仍有 30 bit 的信息无需特别设定，也即为自由位.

由于消息字 $m_0$、$m_2$ 至 $m_4$ 都未在 MD5 的第 2 轮的 $q_{17}$ 之前出现，因此，翻转 $q_1$ 后，我们可以直接修改相应的 $m_1\sim m_4$ 来保持第 1 轮的 $q_2\sim q_5$ 差分路径.

在第 17 步($q_{17}$)中，我们可以通过翻转 $q_1$ 的任意自由位来启动一个新的碰撞对搜索而无需担心影响 $q_{17}$ 之前的差分路径. 而在状态字 $q_{17}$ 之后，dBB 的差分路径有 30 个充分条件需要随机满足，而 $q_1$ 通道也有 30 个自由位，即在 $q_1$ 通道之内能以极大概率找到 1 个 dBB 碰撞对，因此，dBB 碰撞对搜索的起始点就是 $q_{17}$.

应用 $q_1$ 通道之后，根据消息修改技术的原理，$q_1$ 对 MD5 第 1 轮差分路径所产生的影响在 $q_5$ 之后完全消除. 即 $q_1$ 通道的应用，不会影响到 $m_5\sim m_{15}$ 的取

值.因此,可以固定 $m_5 \sim m_{15}$ 的值(352 bit,44 个字符)来对 APOP 的可能密钥字进行猜测.

### 4.2 群满足方案

基于通道技术和高级消息修改技术,确定性地满足在 MD5 碰撞搜索中最后一个通道的首三步的所有充分条件,以达到加速碰撞搜索的目的.

大部分之前的碰撞搜索算法[9,17]都是使用随机满足最后一个通道中的充分条件,但是我们使用群满足方案确定性地满足了首三步的充分条件,因而不仅可以极大地提高碰撞概率而且可以极大提高碰撞对搜索的效率.

### 4.3 应用 $q_9$ 通道猜测密钥的前 11 个字符

我们在 dBB 碰撞搜索中应用分而治之策略和群满足方案,达到控制输入消息最后 96 bit(预先设定的 $m_{13}$、$m_{14}$ 和 $m_{15}$)的目的.相关描述如下.

**基本框架.**

1. 此处的 dBB 碰撞共有 48 个充分条件需要满足(一个额外的条件 $q_{23,22}=1$ 用来支持群满足方案),但我们能够在 $q_{25}$ 步使用完全的 $q_9$ 通道.应用分而治之策略将 dBB 碰撞搜索分成两个阶段:第 1 个阶段从 $q_1 \sim q_{24}$(有 25 个充分条件);第 2 个阶段从 $R_{25}$($q_9$ 通道可被应用)至最后的 $q_{64}$(有 23 个充分条件).

1.1. 从 $q_1$ 到 $q_{13}$,应用简单消息修改技术直接满足充分条件,随机设置非条件位的值.

1.2. 利用预设的 $m_{13} \sim m_{15}$ 的值,依次计算 $q_{14} \sim q_{16}$ 的值,若其中任意条件不满足,转至步 1.1.

1.3. 计算剩余 $m_i$ 值,计算并校验 $q_{17} \sim q_{24}$,若任意条件不满足,转至步 1.1.

2. 应用 $q_9$ 通道群满足方案满足首三步 $q_{25} \sim q_{27}$ 的所有条件(共有 3 个条件),因此第 2 阶段共有 20 个充分条件需要随机满足,$q_9$ 通道有 28 个自由位.

3. 计算并校验 $q_{28} \sim q_{64}$,若任意条件不满足则转至步 2.

在上述框架中的步 2 直接调用了 $q_9$ 通道的群满足方案,接下来在算法 1 中对 $q_9$ 通道的群满足方案进行描述.

**算法 1.** $q_9$ 通道的群满足方案.

输入:dBB 条件差分,消息差分 $\Delta M_1=0$,$q_1 \sim q_{24}$ 的充分条件已满足

输出:$q_{27,31}=q_{26,31}=q_{25,31}=q_{0,31}$

1. 随机设置 $q_9$ 自由位的值,根据 $q_{10}$ 重新计算 $m_9$ 的值.
2. 计算 $q_{25}$ 的值.
3. if $q_{25,31} \neq q_{0,31}$ then
   计算 $q_{9,6}=q_{9,6} \oplus 1$.
   根据 $q_{10}$ 重新计算 $m_9$ 的值.
   计算 $q_{25}$ 的值.
   end if
4. 计算 $q_{26}$ 的值.
5. if $q_{26,31} \neq q_{0,31}$ then
   计算 $q_{9,29}=q_{9,29} \oplus 1$.
   根据 $q_{10}$ 重新计算 $m_9$ 的值.
   计算 $q_{25}$ 和 $q_{26}$ 的值.
   end if
6. 计算 $q_{27}$ 的值.
   if $q_{27,31} \neq q_{0,31}$ then
   计算 $q_{9,24}=q_{9,24} \oplus 1$.
   根据 $q_{10}$ 重新计算 $m_9$ 的值.
   计算 $q_{25}$、$q_{26}$ 和 $q_{27}$ 的值.
   end if
7. if $q_{27,31} \neq q_{0,31}$ 或者 $q_{26,31} \neq q_{0,31}$ 或者 $q_{25,31} \neq q_{0,31}$ then
   转至步 1.
   end if

我们在算法 2 中对应用 $q_9$ 通道生成猜测密钥前 11 位字符的碰撞对进行描述.

**算法 2.** 应用 $q_9$ 通道生成猜测 APOP 密钥前 11 位字符的碰撞对.

输入:dBB 条件差分,消息差分 $\Delta M_1=0$ 以及预先设定的 32 位消息字 $m_{13}$、$m_{14}$ 和 $m_{15}$

输出:$m_0,\cdots,m_{12}$ 和已知的 $m_{13} \sim m_{15}$ 构成 dBB 碰撞

1. for $i=1$ to 13 do
   $q_{i,31} \leftarrow q_{0,31}$, $q_{i,j\neq 31} \leftarrow$ 随机值.
   end for
2. for $i=14$, 16 do
   根据已知 $m_{i-1}$ 的值,计算 $q_i$ 的值.
   if $q_{i,31} \neq q_{0,31}$ then
   转至步 1.
   end if
   end for
3. 计算 $m_1$、$m_6$、$m_{11}$、$m_0$、$m_5$、$m_{10}$ 和 $m_4$ 的值.
4. for $i=17$ to 24 do
   计算 $q_i$ 的值.
   if $q_{i,31} \neq q_{0,31}$ 或者($i=23$ 且 $q_{23,22}=0$) then
   转至步 1.
   end if
   end for
5. 计算 $m_2$、$m_3$、$m_7$ 和 $m_9$ 的值.
6. 调用群满足方案的算法 1.
7. 根据 $q_9$ 计算 $m_8$ 的值.
8. 根据 $q_{13}$ 计算 $m_{12}$ 的值.
9. for $i=28$ to 64 do
   计算 $q_i$ 的值.
   if ($i \notin r_3$)且($q_{i,31} \neq q_{0,31}$) then
   转至步 6.
   end if
   end for
10. 输出 $m_0,\cdots,m_{12}$.

**复杂度分析.** 应用分而治之策略后,算法 2 第 1 阶段和第 2 阶段分别有 12 和 20 个充分条件需要随机满足.由于应用了 $q_9$ 通道,碰撞搜索的起点从 $q_1$

移至 $q_{25}$(在该情形下,一旦算法进入了第 2 阶段,我们只需在 $q_{25}$ 开始碰撞搜索,而无需再从第 1 阶段重新开始).根据分而治之策略,两个阶段的复杂度可以独立计算.第 1 阶段的复杂度大约为 $2^9$ 次 MD5 散列计算,第 2 阶段的复杂度为 $2^{18}$ 次 MD5 散列计算.因此,该算法的总体复杂度为 $2^9+2^{18}\approx 2^{18}$ 次 MD5 计算.应用普通的 PC 机(Intel 2 GHz CPU),具有 96 位信息确定的碰撞对以平均时间 0.08 s 产生,这比 Sasaki 等人[12]的工作要快 $2^4$ 倍.

应用分而治之策略和群满足方案,生成"IV 桥"的复杂度降至 $2^{19}$ 次 MD5 散列计算,平均计算时间约为 0.17 s.由于生成用来恢复密钥前 11 个字符的碰撞对需要 $2^{18}$ 次 MD5 散列计算,因此生成对应 1 比特自由碰撞对所需的计算量为 $2^{36}$ 次 MD5 散列计算.我们在表 1 中与前人的工作进行了比较.

**表 1 复杂度比较**

| 工作 | IV 桥(分组块 1) | 分组块 2 | 1 比特自由碰撞 |
|---|---|---|---|
| 文献[10] | $2^{30}$ | 5 s(3 字符) | — |
| 文献[11] | $2^{30}$ | 5 s(3 字符) | — |
| 文献[12] | $2^{42}$ | 约 1 s(11 字符) | — |
| 文献[13] | $2^{42}$ | 约 1 s(11 字符) | $2^{46}$ |
| 本文 | $2^{19}$ | 0.08 s(11 字符) | $2^{36}$ |

### 4.4 应用 $q_1$ 通道猜测密钥的前 43 个字符

我们在 dBB 碰撞搜索中应用分而治之策略和群满足方案,在 $q_{17}$ 应用 $q_1$ 通道达到控制输入消息最后 352 bit(预先设定的 $m_5\sim m_{15}$)的目的.

我们在算法 3 中对 $q_1$ 通道的群满足方案进行详细描述.

**算法 3.** $q_1$ 通道的群满足方案.

输入:dBB 条件差分,消息差分 $\Delta M_1=0$,$q_1\sim q_{16}$ 的充分条件已满足

输出:$q_{19,31}=q_{18,31}=q_{17,31}=q_{0,31}$

1. 随机设置 $q_1$ 自由位的值,根据 $q_2$ 重新计算 $m_1$ 的值.
2. 计算 $q_{17}$ 的值.
3. if $q_{17,31}\neq q_{0,31}$ then
   计算 $q_{1,6}=q_{1,6}\oplus 1$.
   根据 $q_2$ 重新计算 $m_1$ 的值.
   计算 $q_{17}$ 的值.
   end if
4. 计算 $q_{18}$ 的值.
5. if $q_{18,31}\neq q_{0,31}$ then
   计算 $q_{1,29}=q_{1,29}\oplus 1$.
   根据 $q_2$ 重新计算 $m_1$ 的值.
   计算 $q_{17}$ 和 $q_{18}$ 的值.
   end if
6. 计算 $q_{19}$ 的值.
7. if $q_{19,31}\neq q_{0,31}$ then
   计算 $q_{1,24}=q_{1,24}\oplus 1$.
   根据 $q_2$ 重新计算 $m_1$ 的值.
   计算 $q_{17}$、$q_{18}$ 和 $q_{19}$ 的值.
   end if
8. if $q_{17,31}\neq q_{0,31}$ 或者 $q_{18,31}\neq q_{0,31}$ 或者 $q_{19,31}\neq q_{0,31}$ then 转至步 1.

end if

我们在算法 4 中对应用 $q_1$ 通道产生恢复前 43 个字符的碰撞对进行描述.

**算法 4.** 应用 $q_1$ 通道产生恢复 APOP 密钥前 43 个字符的碰撞对.

输入:dBB 条件差分,消息差分 $\Delta M_1=0$ 以及预先设定的 32 位消息字 $m_5\sim m_{15}$

输出:$m_0,\cdots,m_4$ 和已知的 $m_5\sim m_{15}$ 构成 dBB 碰撞

1. for $i=1$ to 5 do
   $q_{i,31}\leftarrow q_{0,31}$,$q_{i,j\neq 31}\leftarrow$ 随机值.
   end for
2. for $i=6$ to 16 do
   根据已知 $m_{i-1}$ 的值,计算 $q_i$ 的值.
   if $q_{i,31}\neq q_{0,31}$ then
   转至步 1.
   end if
   end for
3. 根据 $q_2$ 计算 $m_1$ 的值.
4. 调用群满足方案算法 3.
5. 根据 $q_1$ 计算 $m_0$ 的值.
6. 根据 $q_3$ 计算 $m_2$ 的值.
7. 根据 $q_4$ 计算 $m_3$ 的值.
8. 根据 $q_5$ 计算 $m_4$ 的值.
9. for $i=18$ to 64 do
   计算 $q_i$ 的值.
   if($i\notin r_3$)且($q_{i,31}\neq q_{0,31}$) then
   转至步 4.
   end if
   end for
10. 返回 $m_0,\cdots,m_4$.

**复杂度分析.** 应用分而治之策略后,算法 4 的第 1 阶段和第 2 阶段分别有 11 和 30 个充分条件需要随机满足.由于应用了 $q_1$ 通道,碰撞搜索的起点从 $q_1$ 移至 $q_{17}$.根据分而治之策略,两个阶段的复杂度可以独立计算.第 1 阶段的复杂度大约为 $2^9$ 次 MD5 散列计算,第 2 阶段的复杂度约为 $2^{26}$ 次 MD5 散列计算.因此,该算法的总体复杂度为 $2^9+2^{26}\approx 2^{26}$ 次 MD5 计算.应用普通的 PC 机(Intel 2 GHz CPU),具有 352 位信息确定的碰撞对以平均时间 20 s 产生,这比 Sasaki 等人[12]的工作要快 $2^{15}$ 倍.

我们也极大改进了用来恢复 APOP 密钥不同长度字符碰撞对的搜索效率.在 $q_{24}$ 中应用 $q_4$ 通道,

能够以 $2^{19}$ 次 MD5 散列计算产生具有 256 bit 确定值的碰撞对(恢复密钥前 31 个字符)，其平均时间约为 0.15 s. 在 $q_{20}$ 中应用我们提出的 $q_1'$ 通道，能够以 $2^{24}$ 次 MD5 散列计算产生具有 320 位信息确定的碰撞对(恢复密钥前 39 字符)，其平均时间约为 4.13 s. 在 $q_{17}$ 中应用我们提出的 $q_1$ 通道，能够以 $2^{26}$ 次 MD5 散列计算产生具有 352 位信息确定的碰撞对(恢复密钥前 43 个字符)，其平均时间约为 20 s. 若在第一个分组块中("IV 桥")的最后 4 个字符应用消息位控制技术(针对 $m_{15}$ 的消息控制不会影响通道和分而治之的使用，所以复杂度并未增加)，我们能够在理论上恢复密钥的前 67 个字符. 针对具有多位信息确定的碰撞对搜索，我们在表 2 中与之前工作进行了比较.

**表 2　具有多位信息确定的碰撞对搜索比较**

| 方案 | 平均时间 | | 复杂度 | | | | | 理论恢复字符/个 |
|---|---|---|---|---|---|---|---|---|
| | 11 字符 | 31 字符 | 35 字符 | 39 字符 | 43 字符 | 47 字符 | 51 字符 | |
| 文献[12] | 约 1 s | 5.86 s | $2^{38}$ | $2^{39}$ | $2^{41}$ | $2^{42}$ | $2^{43}$ | 61 |
| 本文 | 0.08 s | 0.15 s | $2^{24}$ | $2^{24}$ | $2^{26}$ | $2^{39}$ | $2^{40}$ | 67 |

## 5　APOP 的快速密钥恢复攻击

为了实现 APOP 的快速密钥恢复攻击，我们首先伪装成邮件服务器，一旦用户发起连接请求，则从已产生的 MD5 碰撞对中提取挑战数据并将其发送至用户. 为了避开挑战数据的限制条件，我们将挑战数据分为 3 个部分，每部分为 512 bit 的分组块，如下所示.

(1) 分组块 *pre* 以字符'<'开始并只包含一个字符'@'.

(2) 分组块 *mid* 充当"IV 桥"的角色.

(3) 分组块 *end* 包含此次猜测密钥字的相关信息 *cand*，且 *cand* 附加于'>'之后.

我们从碰撞对中导出一个挑战数据对 $(C,C')$，其挑战数据中 $C=pre\parallel mid\parallel(end-cand)$ 而 $C'=pre\parallel mid'\parallel(end-cand)$. 在 *mid* 和 $mid'$ 之间仅有比特 $m_{8,31}$ 的值不同，满足 $m_{8,31}=\neg m'_{8,31}$ 的关系. 如果邮件用户接收到 $C$ 和 $C'$，将当成两个完全不同的挑战. 因此，我们可以使用 $C$ 和 $C'$ 来进行 APOP 密钥恢复攻击的密钥字猜测.

在表 3 中，给出了一个由 3 个分组块组成用来恢复 APOP 密钥前 43 个字符的碰撞对实例. 可能密钥字 *cand* 位于第 3 个分组块 *end* 的尾部，其内容为选自"Bible"开头的"In the beginning God created the heaven and"的 43 个字符. 第 1 个分组块 *pre* 的 64 个字符内容为"<The Holy Bible. King James Version. The Old Testament Gennesis@". 第 2 个分组块 *mid* 是一个精确的"IV 桥"，其为 dBB 提供链接变量差分 $\Delta H_2=-(2^{31},2^{31},2^{31},2^{31})$. 需要指出的是，构造该碰撞对所需的时间约为 20 s.

**表 3　用来恢复 APOP 密钥前 43 个字符的碰撞对实例**

| | | | | |
|---|---|---|---|---|
| $H_0$ | 0x67452301 | 0xefcdab89 | 0x98badcfe | 0x10325476 |
| *pre* | 0x6568543c | 0x6c6f4820 | 0x69422079 | 0x2e656c62 |
| | 0x6e694b20 | 0x614a2067 | 0x2073656d | 0x73726556 |
| | 0x2e6e6e69 | 0x65685420 | 0x646c4f20 | 0x73655420 |
| | 0x656d6174 | 0x4720746e | 0x656e6e65 | 0x40736973 |
| $H_1=H_1'$ | 0x6a8502b | 0xa9db20dd | 0x822f4299 | 0x867a7298 |
| *mid* | 0x47409de4 | 0xa1b6eced | 0x7e70ed35 | 0xd8215bf0 |
| | 0xd7b040e1 | 0x7342a3a5 | 0x79d9e8b3 | 0xe707954f |
| | 0xac618677 | 0x388dc5b7 | 0xf7b86a0c | 0xa040309a |
| | 0x3a617c44 | 0x3392ff6c | 0xf803241d | 0x4f3af7a7 |
| $mid'$ | 0x47409de4 | 0xa1b6eced | 0x7e70ed35 | 0xd8215bf0 |
| | 0xd7b040e1 | 0x7342a3a5 | 0x79d9e8b3 | 0xe707954f |
| | 0x2c618677 | 0x388dc5b7 | 0xf7b86a0c | 0xa040309a |
| | 0x3a617c44 | 0x3392ff6c | 0xf803241d | 0x4f3af7a7 |
| $H_2$ | 0xe6fee770 | 0x21018304 | 0x15c61b85 | 0x7151e9e2 |
| $H_2'$ | 0x66fee770 | 0xa1018304 | 0x95c61b85 | 0xf151e9e2 |
| *end* | 0x8d6f618 | 0xb5db76d5 | 0xa45975e8 | 0x2841c42f |
| | 0x633f5fa5 | 0x206e493e | 0x20656874 | 0x69676562 |
| | 0x6e696e6e | 0x6f472067 | 0x72632064 | 0x65746165 |
| | 0x68742064 | 0x65682065 | 0x6e657661 | 0x646e6120 |
| $H_3=H_3'$ | 0x2665f3c8 | 0x1aa6bb7d | 0x78a0fc27 | 0x2d485e7c |

### 5.1　APOP 本地快速密钥恢复攻击

我们在本地发起一个 APOP 快速密钥恢复攻击，该实验基于一个支持 APOP 认证协议的开源软件"Evolution"的较老版本，我们在软件代码级修改了自动邮件访问的条件，将一般由用户或者计时器发起的访问改为从特定端口收到一个信号发起.

我们在本地 PC(Intel 2 GHz CPU)伪装成邮件服务器($S$)，$U$ 代表邮件用户. 针对 APOP 的逐字节密钥恢复攻击步骤描述如下.

1. $S$ 从包含可能密钥字 *cand* 的碰撞对数据中抽取挑战数据对 $C$ 和 $C'$，发送一个邮件访问信号至用户 $U$.

2. 一旦收到信号，$U$ 尝试连接 $S$.

3. $S$ 发送挑战数据 $C$ 至 $U$.

4. $U$ 计算 $R=MD5(C\parallel passwd)$，并将其发送至 $S$.

5. 一旦接收到 $R$，$S$ 发送另一邮件访问信号至 $U$.

6. 基于收到的信号，$U$ 尝试连接 $S$.

7. $S$ 发送另一个挑战数据 $C'$ 至 $U$.

8. $U$ 计算 $R'=MD5(C' \| passwd)$ 并将其返回至 $S$.

9. $S$ 校验 $R$ 是否和 $R'$ 相等，若是，则此次猜测的可能密钥字 *cand* 正确，若密钥已经完全恢复，则停止. 否则，对密钥可能字 *cand* 进行下一轮的猜测.

**怎样判断密钥已经完全恢复.** 在如上 APOP 密钥恢复攻击的步 9，需要判断密钥是否已经完全恢复. 我们将这个问题解决如下：

(1) 总是先假定密钥 *passwd* 已经完全恢复，则输入消息 $C$ 和 *passwd* 之后的信息就是填充数据 *padding*.

(2) 计算 $MD5(C \| cand)$ 并比较其是否与 $R$ 相等，若是，则确认密钥已经完全恢复，攻击停止. 否则，密钥仍有未知的密钥字有待进一步地恢复.

**实际性分析.** 在实验中，假定本地恶意软件能够修改邮件客户端的行为，这在实际生活中是普遍存在的. 此外，绝大部分的用户习惯勾选记住密钥的选项，省却每次输入密钥的麻烦. 所以该实验一定程度上模拟了由恶意软件发起的攻击，并将该攻击"放大化"处理. 我们还假定密钥仅由 52 个字母和 10 个数字组成，该假设能反映真实生活中一般密钥的取值范围.

一般来说，7 位以上的密钥被认为是安全的. 在攻击中，我们可以快速地恢复出具有 6～31 个字符的密钥，而且能够在实际时间内恢复长达 43 位的密钥. 比如我们能够在大约 1 min 的时间内恢复长达 11 个字符的密钥，不同长度字符密钥恢复攻击所需的平均时间如表 4 中所示.

**表 4 恢复具有不同长度字符的密钥所需平均时间**

| 长度 | 时间/s |
|---|---|
| 6 个字符 | 30.7 |
| 7 个字符 | 35.7 |
| 8 个字符 | 40.9 |
| 9 个字符 | 45.8 |
| 10 个字符 | 51.0 |
| 11 个字符 | 56.0 |
| 31 个字符 | 251.4 |

APOP 的本地快速密钥恢复攻击表明 APOP 的安全性极低，且其它基于 MD5 的应用需要严谨的安全评估.

## 6 总 结

我们提出两个新的通道以恢复 APOP 密钥的更多字符，并提出一种"群满足方案"极大降低了具有多位信息确定的 MD5 碰撞对搜索的复杂度；产生一个用来恢复 APOP 密钥前 31 个字符的碰撞对的平均时间比文献[12]中的工作要快 $2^5$ 倍；产生一个用来恢复 APOP 中密钥的前 39 个字符和 43 个字符碰撞对的平均时间比文献[12]中的工作要快 $2^{15}$ 倍. 这些技术还可以用来降低搜索包含 APOP 密钥的前 11 个字符的 1 比特自由的碰撞对的复杂度.

## 参 考 文 献


[1] Myers J, Rose M. Post office protocol- version 3. RFC 1939 (Standard), 1996. Updated by RFCs 1957, 2449

[2] Preneel B, van Oorschot P C. On the security of two MAC algorithms//Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'96). Saragossa, Spain, 1996: 19-32

[3] Rivest R. The MD5 message-digest algorithm. RFC 1321, 1992

[4] Eastlake D, Jones P. US secure hash algorithm 1 (SHA1). RFC 3174, Internet Engineering Task Force, 2001

[5] Wang Xiao-Yun, Yu Hong-Bo. How to break MD5 and other hash functions//Cramer R ed. Advances in Cryptology (EUROCRYPT 2005). Lecture Notes in Computer Science 3494. Berlin/Heidelberg: Springer, 2005: 19-35

[6] Wang Xiao-Yun, Lai Xue-Jia, Feng Deng-Guo, Chen Hui, Yu Xiu-Yuan. Cryptanalysis of the hash functions MD4 and RIPEMD//Cramer R ed. Advances in Cryptology (EUROCRYPT 2005). Lecture Notes in Computer Science 3494. Berlin/Heidelberg: Springer, 2005: 1-18

[7] Wang Xiao-Yun, Yu Hong-Bo, Yin Yi-Qun. Efficient collision search attacks on sha-0//Shoup V ed. Advances in Cryptology (CRYPTO 2005). Lecture Notes in Computer Science 3621. Berlin/Heidelberg: Springer, 2005: 1-16

[8] Wang Xiao-Yun, Yin Yi-Qun, Yu Hong-Bo. Finding collisions in the full SHA-1//Shoup V ed. Advances in Cryptology (CRYPTO 2005). Lecture Notes in Computer Science 3621. Berlin/Heidelberg: Springer, 2005: 17-36

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

[10] Leurent G. Message freedom in MD4 and MD5 collisions: Application to APOP//Proceedings of the Fast Software Encryption, 14th International Workshop, FSE 2007. Luxembourg, Luxembourg, 2007: 309-328

[11] Sasaki Y, Yamamoto G, Aoki K. Practical password recovery on an MD5 challenge and response. Cryptology ePrint Archive, Report 2007/101, 2007. http://eprint.iacr.org/

[12] Sasaki Y, Wang Lei, Ohta K, Kunihiro N. Security of MD5 challenge and response: Extension of APOP password recovery attack//Proceedings of the 2008 the Cryptographers'

Track at the RSA conference on Topics in cryptology (CT-RSA'08). San Francisco, CA, USA, 2008: 1-18

[13] Wang Lei, Sasaki Y, Sakiyama K, Ohta K. Bit-free collision: Application to APOP attack//Proceedings of the 4th International Workshop on Security: Advances in Information and Computer Security (IWSEC'09). Toyama, Japan, 2009: 3-21

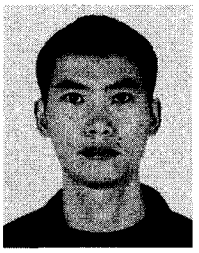
[14] den Boer B, Bosselaers A. Collisions for the compression function of MD5//Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology (EUROCRYPT'93). Secaucus, NJ, USA, 1994: 293-304

[15] Liu Fan-Bao, Xie Tao, Feng Yu-Meng, Feng Deng-Guo. On the security of PPPoE network. Security and Communication Networks, 2012: 1-10

[16] Liu Fan-Bao. On the security of digest access authentication//Proceedings of the 14th IEEE International Conference on Computational Science and Engineering, 11th International Symposium on Pervasive Systems, Algorithms, and Networks, and 10th IEEE International Conference on IEEE International Conference on Ubiquitous Computing and Communications (IUCC 2011). Dalian, Liaoning, China, 2011: 427-434

[17] Stevens M. On collisions for MD5 [M. S. dissertation]. Eindhoven North Brabant, Netherlands: TU Eindhoven, Faculty of Mathematics and Computer Science, 2007

[18] Xie Tao, Liu Fan-Bao, Feng Deng-Guo. Could the 1-MSB input difference be the fastest collision attack for MD5? Cryptology ePrint Archive, Report 2008/391, 2008. http://eprint.iacr.org/



**LIU Fan-Bao**, born in 1982, Ph. D. candidate. His main research interests include cryptanalysis, network security and trusted computing.

**XIE Tao**, born in 1966, Ph. D., professor. His research interests include evolutionary computation, cryptography and information security.

**FENG Deng-Guo**, born in 1965, Ph. D., professor. His research interests include network and information security, cryptography theory and technique.

**Background**

Evaluating the security of some prevalent hash functions and their applications is a hot issue nowadays, since hash function is a fundamental cryptographic primitive. The collision resistance of MD5 and SHA-1, which are both the most widely applied hash functions, is broken practically and theoretically, respectively. Special attentions have been focused on applying the MD5 collision pair to break the APOP (Authentication Post Office Protocol) system, for the reason of that mail systems include a lot of personal information. The best attack before this attack can generate the collision pair for recovering the first 11 characters in about one second. This paper proposes a new password recovery attack to APOP, which can recover more password characters and recover passwords faster. Our results show that APOP cannot be used any more.

This work was supported by the program "Core Electronic Devices, High-End General Purpose Chips and Basic Software Products" in China (No. 2010ZX01037-001-001). This work was supported by National Basic Research Program (973 Program) of China (No. 2007CB311202). This work was supported by the National Natural Science Foundation of China (No. 61070228).