

基于彩虹表的时间-存储折中攻击改进算法^{*}

郑中翔¹, 吉庆兵², 于红波¹

1. 清华大学 计算机系密码理论与技术研究中心, 北京 100084

2. 保密通信重点实验室, 成都 610041

通讯作者: 于红波, E-mail: yuhongbo@mail.tsinghua.edu.cn

摘要: 杂凑函数是将任意长度的字符串映射到固定长度输出的函数, 由于其具有单向性而被广泛的应用于口令认证。许多网站或服务器都使用杂凑函数来保存用户口令。时间-存储折中攻击是 1980 年由 Martin Hellman 提出的, 它能够在有限的存储和计算能力的限制下, 在可接受的时间内计算出常用的计算机口令。而 Philippe Oechslin 在 2003 年提出的彩虹表法极大地提升了时间-存储折中攻击的效率, 此后在彩虹表法的基础上, 大量改进算法被纷纷提出。本文提出了一种基于彩虹表的时间存储-折中攻击改进算法, 它应用了概率统计的方法, 能够在基本保证成功率的基础上大大提升搜索效率。即当成功率降低 4.12% 时, 搜索时间降低 86.21%, 它是一种效率与成功率折中的算法。

关键词: 时间-存储折中; 彩虹表; LM-Hash; NT-Hash; 概率函数

中图法分类号: TP309.7 **文献标识码:** A

中文引用格式: 郑中翔, 吉庆兵, 于红波. 基于彩虹表的时间-存储折中攻击改进算法[J]. 密码学报, 2014, 1(1): 100–110.

英文引用格式: Zheng Z X, Ji Q B, Yu H B. Faster cryptanalytic time-memory trade-off using rainbow table[J]. Journal of Cryptologic Research, 2014, 1(1): 100–110.

Faster Cryptanalytic Time-memory Trade-off Using Rainbow Table

ZHENG Zhong-Xiang¹, JI Qing-Bing², YU Hong-Bo¹

1. Center for Cryptology Study, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
2. Science and Technology on Communication Security Laboratory, Chengdu 610041, China

Corresponding author: YU Hong-Bo, E-mail: yuhongbo@mail.tsinghua.edu.cn

Abstract: A cryptographic hash function is a function that takes an arbitrary-length data and returns a fixed-size digest. Hash functions are widely used in password authentication based on the one way property. Many websites or servers save user passwords using a hash function. Time-memory trade-off attack was presented by Martin Hellman in 1980. With limited storage and computing capacity, the attacker can get the passwords in an acceptable period of time. In 2003, Philippe Oechslin proposed a more efficient method by introducing a new table structure called rainbow table. Then, a number of improvements have been proposed based on rainbow table. In this paper, we propose a faster cryptanalytic time-memory trade-off using rainbow table based on probability statistics, which can greatly improve the searching efficiency with small losses of success rate. The searching time will be reduced 86.21% compared with the original algorithm when the success rate has a loss of 4.12%. The algorithm is a trade-off between efficiency and success rate.

* 基金项目: 国家重点基础研究发展计划(973 计划)(2013CB834205); 国家自然科学基金(61133013, 61373142); 清华大学自主科研计划(20111080970); 保密通信重点实验室基金项目(9140C110404110C1106)

收稿日期: 2013-12-15 定稿日期: 2013-12-26

Key words: time-memory trade-off; rainbow table; LM-Hash; NT-Hash; probability function

1 引言

杂凑(Hash)函数^[1]是一类基础密码算法, 它能将任意长度的输入映射到固定长度的输出, 具有单向性(抗原像攻击)、抗第二原像攻击、抗碰撞三类安全属性, 被广泛应用于口令认证、数据完整性检测、数字证书等方面。口令认证基于杂凑函数的单向性(抗原像攻击的属性)。对于需要保存在本地计算机或服务器的口令, 明文保存存在很大的风险(如 2011 年 CSDN 发生的口令泄露事件)。常用的方法是保存口令的杂凑值, 当用户输入口令时, 服务器计算其杂凑值, 与存储在系统中的杂凑值相比较, 并依此判断用户的输入是否正确。

LM-Hash(LAN Manager Hash)^[2]是微软用于保存网络操作系统以及 Windows 操作系统口令而设计的杂凑函数。这个函数基于 DES 算法, 但设计上存在一定的缺陷, 微软在 Windows Vista 及之后的版本不再使用这个杂凑函数作为默认的口令存储函数, 然而这个函数在研究时间-存储折中攻击时仍然具有典型的意义。NTLM-Hash^[3](简称 NT-Hash)主要应用于 Windows 的网络环境中, 它基于杂凑函数 MD4, 这个杂凑函数最早在 Windows NT 4.0 SP4 中被广泛使用, 随后又被应用于 Windows 2000 中, 虽然已经被 Kerberos^[4]取代而不再作为默认的认证协议, 但仍然在很多没有采用 Kerberos 协议的服务器和客户端通信中被使用。

对登陆口令破解最常用的方法是时间-存储折中攻击(TMTO)。时间-存储折中攻击是 1980 年由 Martin Hellman 提出的^[5], 该方法分为预算过程和在线密钥(或口令)检索两个阶段。在预算阶段, 通过迭代系列起始点建立密钥(或口令)的链表, 构建 TMTO 表来存储链表的头结点和尾结点, 供在线检索使用。在线密钥(或口令)检索阶段是对给定的密文(或口令的杂凑值), 通过检索 TMTO 表, 来寻找对应的密钥(或口令)。TMTO 方法通过预算来达到在查找过程时间和存储的折中。这种时间-存储折中的方法是一种概率模型, 无法确保 100% 成功率, 成功率的大小将取决于分配的时间和内存的大小。1982 年, Ronald L. Rivest 在时间存储折中攻击中引进了区分点的概念从而大幅减少在分析过程中的内存查找^[6], 之后的大量研究都在区分点的方法上改进和提高。直到 2003 年 Philippe Oechslin 在文献[7]中选择了一条与区分点法不同的道路——彩虹表法, 这一方法在提供区分点法的各项优点的同时避免了区分点法链长不固定带来的缺陷, 作为一种可以替代区分点法的新思路受到了广泛的关注, 此后的很多研究都基于彩虹表法, 比较典型的有检查点法^[8]、常用字典攻击^[9]等。

本文基于彩虹表的研究工作, 分析了彩虹表中数据分布的一般规律, 并根据这个规律提出了基于概率统计的彩虹表改进算法, 通过在彩虹表的查找过程中应用概率函数, 实现了成功率与查找时间的折中, 即能够在保证较高查找成功率的基础上大幅减少查找时间, 当成功率由原方法的 95.88% 时, 搜索时间降低为原方法的 13.79%, 我们通过实验证明了我们提出的改进算法的有效性。在实际应用中, 彩虹表常用于攻击数据库中大量口令的摘要值, 这时更关注查找效率。因此本文提出的牺牲少量成功率的基础上大幅降低检索时间的方法具有重要的实际意义。

2 预备知识

2.1 LM-Hash与NT-Hash

LM-Hash 是基于分组密码 DES 实现的, 其运算过程主要有以下 6 个步骤:

- (1) 将口令中所有的小写字母转换为大写字母;
- (2) 将口令长度固定为 14 字节, 若口令长于 14 字节则截取前面的 14 字节, 若不足 14 字节, 则用空字符补足空余的字节;
- (3) 将得到的 14 字节数据分为前后两组 7 字节(56 bit)数据;

- (4) 将两组 56 bit 的数, 分别在每 7 bit 后插入一个空比特来生成 64 bit 的 DES 算法密钥值 K_1, K_2 ;
- (5) 用 K_1, K_2 作为密钥值分别对明文 “KGS!@#\$%” 做 DES 加密, 获得两组 64 bit 的密文 C_1, C_2 ;
- (6) 级联 C_1, C_2 得到 128 bit 的 LM-Hash 输出摘要.

NT-Hash 是基于杂凑函数 MD4 实现的, 其运算过程为:

- (1) 将口令的标准 ASCII 串按小端(little-endian)序转换为 Unicode 串;
- (2) 对获得的 Unicode 串使用标准 MD4 计算摘要(128 bit), 作为 NT-Hash 的输出.

2.2 时间-存储折中攻击(TMTO)

对固定的明文 P 和某个密文 C_0 , 已知加密算法 S , 目标是恢复出加密过程中使用的密钥 K_0 :

$$C_0 = S_{K_0}(P)$$

针对这个问题, Martin Hellman 在 1980 年提出了时间-存储折中攻击(TMTO)的方法^[5], 这个方法主要由预计算和在线检索两个部分组成.

预计算:

预计算过程的主要工作是构造一个容纳尽可能多密钥的表, 首先随机选取一定数量的密钥, 分别将这些密钥通过交替应用加密函数 S 和约减函数 R , 生成若干条密钥和密文交替出现的链, 如式(1)所示:

$$K_i \xrightarrow{S_{K_i}(P)} C_i \xrightarrow{R(C_i)} K_{i+1} \xrightarrow{S_{K_{i+1}}(P)} C_{i+1} \xrightarrow{R(C_{i+1})} K_{i+2} \quad (1)$$

令函数 $f(K_i) = R(S_{K_i}(P_0))$, 则 $f(K_i)$ 为从密钥到密钥的映射. 使用 $f(K_i)$, 可将式(1)写成以下密钥链的形式, 如式(2):

$$K_i \xrightarrow{f(K_i)} K_{i+1} \xrightarrow{f(K_{i+1})} K_{i+2} \xrightarrow{f(K_{i+2})} \dots \quad (2)$$

构造 m 条长度为 n 的链, 并将它们的头尾元素储存在一个表中, 这样就建立了一张密钥表, 如图 1.

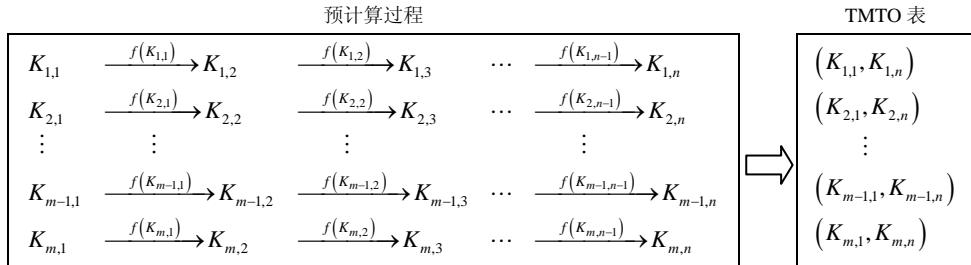


图 1 TMTO 表建立过程
Figure 1 Establishment of a TMTO table

在线检索:

在线检索密钥表的过程中, 对任意的密文 C_i , 如果与它相对应的密钥被包含在上述密钥表中, 那么可以通过以下方式检索出该密钥.

- (1) 令 $K_i = R(C_i)$, 将 K_i 与 $K_{1,n}, K_{2,n}, \dots, K_{m,n}$ 分别比较;
- (2) 若 K_i 恰好与某 $K_{j,n}$ 相同, 则通过对 $K_{j,1}$ 连续使用 $n-2$ 次 $f(K_i)$ 函数得到 $K_{j,n-1}$, 有 $S_{K_{j,n-1}}(P) = C_i$, $K_{j,n-1}$ 即为目标密钥;
- (3) 若 K_i 与 $K_{1,n}, K_{2,n}, \dots, K_{m,n}$ 均不相同, 则求 $K_{i+1} = f(K_i)$, 将 K_{i+1} 与 $K_{1,n}, K_{2,n}, \dots, K_{m,n}$ 比较, 重复上述过程直至找到目标密钥或遍历密钥表.

在建立密钥表时, 不同的链之间可能会生成相同的元素, 称为碰撞; 而当碰撞产生后, 这两条链将生成完全相同的元素, 这种情况称为合并。这两种现象会使密钥表产生大量重复的元素从而影响检索效率。另外, 由于约减函数对输入和输出空间的大小没有要求, 因此可能存在将不同的输入映射到相同的输出, 在检索过程中可能会引起不必要的计算, 这种情况称为错误警报。为解决以上问题, TMTO 法同时使用多张密钥表, 这些密钥表中使用的约减函数不同, 这样的多张小表中出现碰撞和合并的概率比仅使用一张大表时低得多, 因此这一方法提高了检索效率和成功率。

2.3 彩虹表

TMTO 法避免了不同密钥表间出现碰撞和合并, 但同一张表中依然可能存在合并和碰撞, 且这些合并和碰撞仍然是检索表时额外计算开销的主要原因。2003 年, Philippe Oechslin^[7]提出的彩虹表法成功解决了这一问题。

在彩虹表法中, 针对同一条密钥链, 彩虹表使用一组约减函数而非一个约减函数, 如图 2。其中 $f_j(K_i) = R_j(S_{K_i}(P_0))$, 在即第一步使用约减函数 R_1 , 第二步使用约减函数 R_2 , ……, 最后使用约减函数 R_{n-1} 。当两个链发生碰撞时, 它们只有在碰撞发生在两条链的同一个位置的时候才会合并。如果碰撞并不在同一个位置发生, 那么两条链的下一次生成将会使用不同的约减函数, 因此它们不会合并。对于长度为 n 的链, 当冲突发生时, 它最终合并的概率是 $1/n$ 。

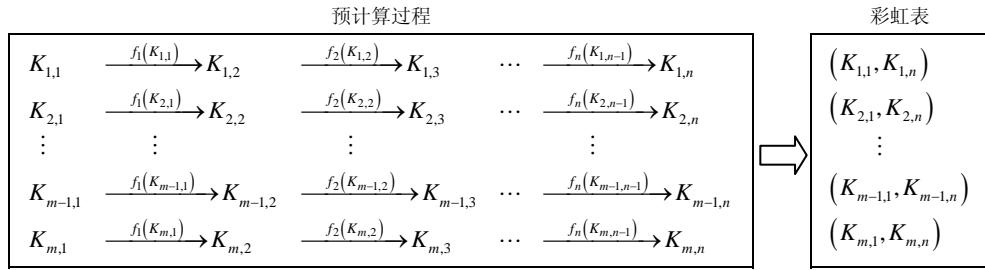


图 2 彩虹表建立过程
Figure 2 Establishment of a rainbow table

为了在彩虹表中查找一个特定的密钥, 可以采取以下的步骤: 首先对密文求 R_{n-1} , 然后在表中的尾元素中查找, 如果找到就可以通过对应的首元素构造该链, 如果没找到, 则可以通过对密文使用 R_{n-2} , f_{n-1} 来看是否有相同的尾元素, 依次类推。这种方式的总计算量为 $n(n-1)/2$, 为原方法的二分之一。实际上, 大概需要进行 n^2 次计算来查找 n 个 $m \times n$ 的表。彩虹表法和 TMTO 法相比, 查表的次数降低了一半, 而且彩虹表中每个约减函数仅出现一次, 因此任意一条链中都没有循环。与其他一些使用非定长链的方法(如区分点法)相比, 彩虹表的链都为固定长度, 这让搜索能够在有限时间内返回结果, 进一步提高效率。

在使用彩虹表攻击杂凑函数时, 与上述攻击加密算法时稍有不同, 因为杂凑函数中仅有消息和摘要, 不含密钥。因此针对杂凑函数的攻击是求原像, 即已知杂凑函数的算法及摘要, 求消息。在建立彩虹表的过程中, 需要设计约减函数 $R(x)$ 为摘要值到消息值的映射, 并以随机消息为起始, 杂凑函数与约减函数 $R(x)$ 为映射关系, 建立一张消息表。同样的, 在检索该表时, 需要比较的对象也由密钥变为消息, 其余各步骤均与攻击加密函数时相同。

彩虹表法改进了 TMTO 法, 大幅提高了检索效率, 因此后来彩虹表法成为了时间-存储折中攻击的首选。同时, Philippe Oechslin 提出了完美表的设想, 即所有链头尾元素均不同, 此后有大量研究工作都在完美表的基础上开展, 如文献[10]。相对于非完美表, 完美表具有更小的存储空间以及更好的查找效率, 但也

需要更多的预算算。完美表的目标是尽可能减少表中碰撞的出现，这能够让表占据的空间更小同时效率更高，但完美表的概念目前仍然难以实现。因此，任一个足够大的彩虹表总包含大量的碰撞，我们提出的基于概率统计的改进算法正是针对这种现象。

3 基于彩虹表的时间-存储折中攻击改进算法的设计与实现

我们提出了一种基于彩虹表的时间-存储折中攻击改进算法，同时整理了在实现算法的具体过程中遇到的一些问题和解决方法，如约减函数的选择，输入输出字符的处理等。

3.1 彩虹表约减函数的选取

在原始的时间-存储攻击中，对约减函数 $R(x)$ 的唯一要求是实现由密文到密钥的映射。但约减函数 $R(x)$ 的好坏能够直接影响表的元素覆盖率，所以，一个选取恰当的约减函数 $R(x)$ 可以大幅度节省时间和空间的开销。

以 LM-Hash 为例，LM-Hash 的输入为 14 个 ASCII 字符，而输出为 16 个 ASCII 字符，即映射的输入空间大小为 95^{14} ，然而约减函数 $R(x)$ 的输入并不需要如此庞大，这是因为在 LM-Hash 的计算中，实际输入的字节数限制在 14 个，超过 14 个字节的部分将被截断，而这 14 个字节将分为前后两个 7 字节分别作为密钥对一个特定的字符串进行 DES 加密，最后将两个输出的 8 字节的串拼接在一起作为输出。因此在时间-存储攻击过程中，只需要建立消息长度为 7 个字节的彩虹表就能够满足整个 LM-Hash 的输入空间。约减函数 $R(x)$ 的输入和输出空间恰与 LM-Hash 函数相对，即约减函数 $R(x)$ 的输入空间为 95^8 。再考虑约减函数 $R(x)$ 的输出空间，由于约减函数 $R(x)$ 需要实现从摘要值到消息值的映射，即 $R(x)$ 的输出应在消息空间。在实际问题中，密钥只能选择 0–9 的数字集，A–Z、a–z 的字母集，以及 20 个常见特殊字符(如\$@#等)，且 LM-Hash 函数具有对大小写字母不敏感的特点(即在处理消息时会将其中所有的字母转为大写)。所以无需考虑字母大小写情况，因此 $R(x)$ 的有效输出空间约为 56^7 。

确定了约减函数 $R(x)$ 的输入输出空间，下面需要确定的是映射关系。需要考虑以下几个条件：

- (1) 能够接受输入空间内的所有输入；
- (2) 能且仅能映射到有效输出空间的所有输出；
- (3) 运算简单快捷；
- (4) 迭代不含循环；
- (5) 迭代不收敛；
- (6) 迭代能尽快扩散到整个空间。

权衡以上 6 个条件，最终本文选择了以下约减函数组 $R(x)=\{R_i(x)|1 \leq i \leq n\}$ ，如式(3)。

$$R_i(x) = g(f(x, i, 0) \bmod 56) \| g(f(x, i, 1) \bmod 56) \| \cdots \| g(f(x, i, l_2 - 1) \bmod 56) \quad (3)$$

其中 x 为约减函数的输入字符串，其字节长度为 l_1 ； y 为约减函数的输出字符串，其字节长度为 l_2 ； $R_i(x)$ 代表函数组中第 i 个约减函数($1 \leq i \leq n$)。 $f(x, i, j)$ 如式(4)所示， $g(\theta)$ 的定义如表 1 所示。其中输入 $\theta \in [0, 55]$ ，对于任意输入 θ ，表 1 第一列代表 θ 的十位数，第一行代表 θ 的个位数， $g(\theta)$ 输出为 θ 对应行列交点中的值，如： $g(34)='y'$ ， $g(47)='+'$ 。

$$f(x, i, j) = (x + i)_{(j) \bmod (l_1)} + (x + i)_{(j+l_2) \bmod (l_1)} + \frac{Z}{i+1} + \gamma \quad (4)$$

表 1 θ 到 $g(\theta)$ 的映射关系
Table 1 Mapping relationship between θ and $g(\theta)$

$g(\theta)$	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	a	b	c	d	e	f	g	h	i	j
2	k	l	m	n	o	p	q	r	s	t
3	u	v	w	x	y	z	!	@	#	\$
4	%	^	&	*	()	-	+	=	[
5]	<	>	/	?					

式(4)中, Z 为一个足够大的整数(例如 10^8), $Z/(i+1)$ 为除法取整运算; γ 代表建立多张小表时, 每张小表的偏移量, 取值为小表的序号($[0,63]$); $(x+i)$ 代表将字符串 x 视为整数, 与整数 i 求和, 然后将加和转换为字符串的运算; $(x+i)_{(j)\text{mod}(l)}$ 代表 $(x+i)$ 字符串中的第 $(j)\text{mod}(l)$ 个字符.

约减函数组 $R(x)$ 中只使用了加法、取模和除法运算, 能够接受输入空间内的全部输入, 同时也能够保证输出均在有效输出空间内. 在约减函数组的构造过程中, 使用一个大整数除以组序号作为偏移量能够很好地达到抵抗循环和收敛的作用.

3.2 彩虹表中数据的概率分布

我们使用 3.1 节中所述的约减函数组 $R(x)$, 以长度为 6 字节的字母、数字和 10 个常用特殊字符 ($!@#$%^&*$) 组成的口令为目标, 口令空间大小为 46^6 , 依据 Martin Hellman 在 1980 年^[5]提出口令空间与建表规模的关系, 选择彩虹表中元素个数为 $64 \times 120000 \times 4000$, 如表 2 所示. 其中, 总表数表示彩虹表中小表的数量; 单表数据规模代表彩虹表中每张表中包含的链数与链长度的乘积; 表格大小指该彩虹表占据存储空间大小.

表 2 用约减函数组 $R(x)$ 建立的彩虹表检索测试结果
Table 2 Test result of the rainbow table established by reduction function group $R(x)$

总表数	单表数据规模	表格大小
64	120000×4000	$1.40 \text{ M} \times 64$

以我们建立的彩虹表为实验对象, 探索迭代深度 t 与覆盖率 R_{cover} 的关系.

定义 1(迭代深度) 迭代深度 t 表示彩虹表链中第 $j (1 \leq j \leq n)$ 列的元素 $K_{i,j}$ 与其所属链上尾元素 $K_{i,n}$ 连同两者之间元素构成链的链长 $n+1-j$ 与总链长 n 的比值, 即位于第表中第 j 列元素 $K_{i,j}$ 的迭代深度为 $(n+1-j)/n$. 迭代深度的取值 $t \in \{1/n, 2/n, \dots, (n-1)/n, 1\}$.

定义 2(覆盖率) 覆盖率 R_{cover} 是迭代深度 t 的函数, 覆盖率表示表格中位于此深度及更浅深度的元素中, 不同元素的个数与整张表中不同元素个数的比值($0 < R_{\text{cover}} \leq 1$).

我们分别测试了针对 LM-Hash 和 NT-Hash 的彩虹表攻击, 经测试得到两表中迭代深度和覆盖率的关系如表 3 所示.

通过对表 3 中的数据做指数拟合^[11], 得到覆盖率和迭代深度的关系可以用式(5)表示:

$$R_{\text{cover}} = A_1 \times e^{B_1 t} + A_2 \times e^{B_2 t} \quad (5)$$

其中 R_{cover} 表示覆盖率, t 表示迭代深度, A_1, A_2, B_1, B_2 为常数, 与约减函数的选取有关.

本实验中 $A_1 = 0.905, A_2 = 0.067, B_1 = -1.280, B_2 = -21.98$. 覆盖率与迭代深度之间的关系如图 3 所示, 其中横坐标为迭代深度, 纵坐标为覆盖率:

表 3 迭代深度与覆盖率的关系(%)
Table 3 Relationship between t and $R_{\text{cover}} (\%)$

迭代深度	覆盖率		迭代深度	覆盖率	
	LM-Hash	NT-Hash		LM-Hash	NT-Hash
10	78	78	60	96	95
20	91	90	70	97	96
30	93	92	80	98	97
40	94	93	90	99	99
50	95	94	100	100	100

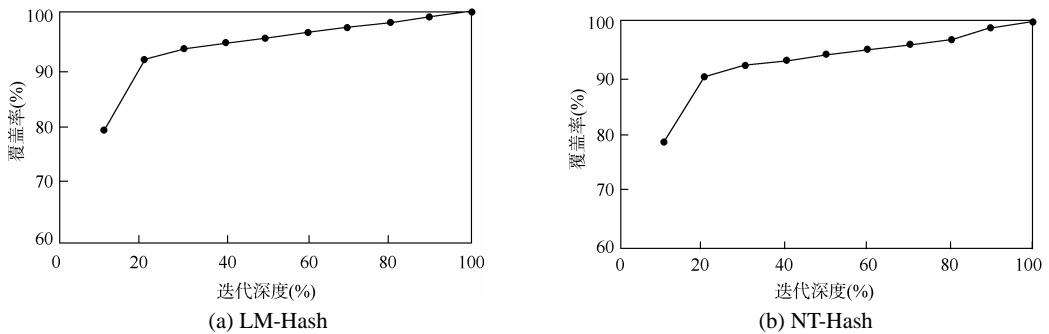


图 3 迭代深度与覆盖率
Figure 3 Relationship between t and R_{cover}

出现这样的迭代深度与覆盖率的关系, 主要原因是彩虹表中存在大量的碰撞, 在建立表的过程中, 随着表规模的增大, 表中包含的元素逐渐增多, 但出现重复元素(即碰撞)的概率也随之增大. 因此覆盖率的增长率随迭代深度的增加而递减, 这使得覆盖率与迭代深度形成了图 3 所示的关系, 可以看出 LM-Hash 与 NT-Hash 覆盖率与迭代深度基本一致, 因此以下的实验结果均选取 LM-Hash 为代表, NT-Hash 的结果也与 LM-Hash 相近.

3.3 基于彩虹表的时间-存储折中攻击改进算法

在时间-存储折中攻击中, 面临的最大挑战就是数据链之间的碰撞与合并, 而时间上的最大开销来自众多的错误警报. 算法上的主要提升都是通过降低碰撞和合并的可能性或者减少错误警报数量来实现的. 如彩虹表法在每一个迭代中使用不同的约减函数大大降低了合并的可能性. 而本文中提出的方法主要是从减少错误警报数量的角度考虑.

在其他条件不变的情况下, 迭代深度增加对覆盖率的提高速率的影响逐渐降低, 而检索时的运算开销却随着迭代深度 t 的增加而以 $O(t^2)$ 的复杂度增加(见文献[7]复杂度的计算). 同时, 基于我们的大量实验

发现, 在彩虹表法中搜索成功时遇到的平均错误警报数大约为搜索失败时遇到的错误警报数的 20%–50%. 在此基础上, 我们选择在搜索表格时以概率函数 $P(t)$ 的值($P(t) \in [0,1]$) 作为可能性判断继续或退出搜索, 其中 $P(t)$ 的取值随着 t 的增加而逐渐降低, 从而达到在大体保证成功率的基础上, 大大降低搜索所需时间的目的. 我们的改进算法应用于表格检索的过程中, 建立彩虹表后, 采用以下算法来查找任意目标消息值:

算法 1 基于彩虹表的时间-存储折中攻击改进算法

1. 输入为目标消息的摘要值 C_i .
 2. 对每一张小表, 取初值 $t = 1/n$, 遵循以下规则:
 - 2.1 以迭代深度 t 为输入, 计算概率函数 $P(t)$ 的值;
 - 2.2 随机选取 $[0,1]$ 间的某数 c ,

若 $c \geq P(t)$, 转 4; 若 $c < P(t)$, 转 2.3;
 - 2.3 在当前迭代深度 t 下, 检索这一深度的元素中是否有目标值(方法与彩虹表法的表格检索相同),

若检索成功, 转 3; 若检索失败, 转 2.4;
 - 2.4 迭代深度 $t \leftarrow t + 1/n$,

若 t 大于 1, 转 4; 否则, 转 2.1.
 3. 输出为目标消息 P_i , 搜索成功, 结束.
 4. 等待所有小表的结果, 若均未成功, 则搜索失败, 结束.
-

根据实验中获得的迭代深度和覆盖率的关系(式(5)), 我们可以计算出改进算法与原彩虹表方法的理论运行时间和成功率, 理论上搜索时间变化如式(6), 成功率的变化如式(7):

$$\frac{\bar{T}}{T} = \frac{\int_0^1 T'(t) \times \prod_{i=0}^t P(i) dt}{\int_0^1 T'(t) dt} \quad (6)$$

$$\frac{\overline{P}_{\text{suc}}}{P_{\text{suc}}} = \frac{\int_0^1 R'_{\text{cover}}(t) \times \prod_{i=0}^t P(t) dt}{\int_0^1 R'_{\text{cover}}(t) dt} \quad (7)$$

其中, T 和 \bar{T} 分别表示应用概率函数之前和之后的理论搜索时间, P_{suc} 和 $\overline{P}_{\text{suc}}$ 分别表示应用概率函数之前和之后的理论成功率, $T'(t)$ 代表对 $T(t)$ 求导运算, $R'_{\text{cover}}(t)$ 代表对 $R_{\text{cover}}(t)$ 求导运算.

选取式(8)为概率分布函数:

$$P(t) = \begin{cases} 1, & t \leq \alpha \\ \alpha/t, & t > \alpha \end{cases} \quad (8)$$

其中, t 为迭代深度, $0 < t \leq 1$; α 为临界比例(当深度超过 α 时, 概率函数取值小于 1, 以一定的概率继续搜索或退出).

当 α 取 50% 时, 可以计算得到 $\overline{P}_{\text{suc}}/P_{\text{suc}} = 97.58\%$, $\bar{T}/T = 59.80\%$.

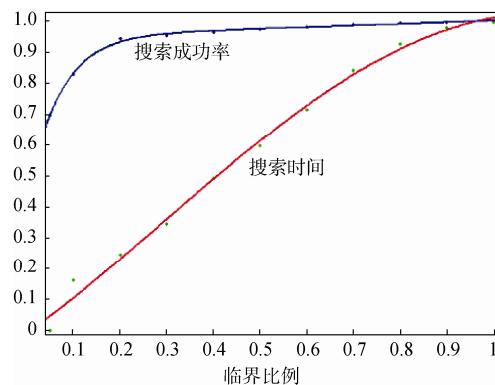
同理可计算得到 α 取其他值时, 理论上的成功率和时间的变化情况如表 4 所示 (其中 α 取 100% 时表示概率函数值恒为 1, 退化为彩虹表法):

表 4 临界比例与理论计算(%)

Table 4 Relationship between $\overline{P}_{\text{suc}}/P_{\text{suc}}$, \bar{T}/T and α by computing (%)

α	$\overline{P}_{\text{suc}}/P_{\text{suc}}$	\bar{T}/T	α	$\overline{P}_{\text{suc}}/P_{\text{suc}}$	\bar{T}/T
10	83.18	16.33	60	98.42	71.58
20	94.56	24.65	70	99.12	84.30
30	95.66	34.59	80	99.62	92.98
40	96.64	49.47	90	99.90	98.10
50	97.58	59.80	100	100.00	100.00

从表 4 的实验结果中可以看出, 当 α 逐渐减小时, 理论成功率降低的速率低于理论时间的降低速率, 因此在实际满足成功率需要的情况下, 可以通过降低 α 的界限来提高效率, 其变化趋势如图 4 所示.

图 4 理论计算中临界比例 α 与成功率和时间的关系Figure 4 Relationship between $\overline{P}_{\text{suc}}/P_{\text{suc}}$, \bar{T}/T and α by computing(abscissa for α , ordinate for ratio)

3.4 实验结果

经过测试得到结果, 上述约减函数能够在 100 M 内存的限制下, 计算随机选取的多组 6 位字母、数字及 10 个常用特殊字符的组合, 达到 75% 以上的成功率, 并且每次搜索的平均时间不超过 1.5 分钟.

以式(8)为概率分布函数, 选取 α 取值分别为 0%–100%(其中 100% 时退化为彩虹表), 实验结果如表 5 及图 5:

表 5 临界比例与实验结果(%)

Table 5 Relationship between $\overline{P}_{\text{suc}}/P_{\text{suc}}$, \bar{T}/T and α by experiments (%)

α	$\overline{P}_{\text{suc}}/P_{\text{suc}}$	\bar{T}/T	α	$\overline{P}_{\text{suc}}/P_{\text{suc}}$	\bar{T}/T
10	85.57	2.59	60	97.94	46.55
20	95.88	7.76	70	97.94	58.62
30	95.88	13.79	80	98.97	72.41
40	96.91	23.28	90	97.94	89.66
50	96.91	33.62	100	100.00	100.00

3.5 数据分析

如图 5 所示, 从总体趋势上可以看出, 随着临界比例 α 的增大, 成功率逐渐提高, 但当 α 增大到 20%

或以上时, 成功率提高效果并不明显; 而搜索时间则随着 α 增大而逐渐增大。这一趋势与理论计算结果(图 4)所示吻合, 但理论上的搜索时间和实验中的搜索时间在数值上有一定偏差, 这是由于理论计算中所考虑的是搜索时间的上界, 因此比实验结果偏高。在误差允许的范围内, 我们认为实验结果证实了理论计算的正确性。

根据以上结论, 概率函数能够帮助我们在少量降低成功率的代价下大幅提升搜索效率。例如, 当成功率为 97% 时, 搜索时间为 580 s, 而若将成功率降为 95%, 则搜索时间减少至 255 s; 若要求在 45 s 时间内完成检索, 仍可以达到 93% 的成功率。

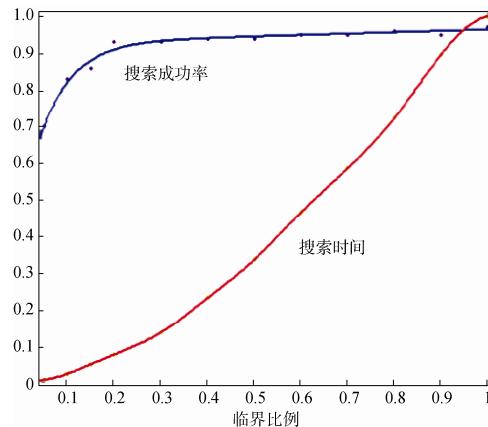


图 5 实验中临界比例 α 与运行时间和成功率

Figure 5 Relationship between $\overline{P}_{\text{suc}}/P_{\text{suc}}$, \bar{T}/T and α by experiments(abscissa for α , ordinate for ratio)

4 总结

本文提出了一种基于彩虹表的时间-存储折中攻击改进算法, 并给出了理论证明和实验结果, 同时本文还提出了约减函数的构造原则并提出了一种性能良好的约减函数, 较好的满足了彩虹表实验的需要。我们提出的基于概率统计的改进算法的最大的优势在于, 他可以灵活的利用表结构而不需要重构整张表, 使得同一张表可以在不同的情况下发挥更好的作用, 也提供了一种研究彩虹表问题的新思路。我们在实验中证明了深度与覆盖率的关系在彩虹表的研究中有很大实际意义, 彩虹表常用于攻击数据库中大量口令的摘要, 检索效率是考虑的主要因素。我们的改进算法能够在牺牲少量成功率的代价下大幅提升检索效率, 满足了实际问题的需要。

时间-存储折中攻击的实施并不依赖于具体的杂凑算法, 因此我们的基于彩虹表的时间-存储折中攻击改进算法对当前通用的杂凑函数如 MD5、SHA-1、SHA-2 等均适用。

References

- [1] Rogaway P, Shrimpton T. Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance[C]. In: Fast Software Encryption. Springer Berlin Heidelberg, 2004: 371–388.
- [2] Burnett M. Ten windows password myths[EB/OL]. <http://www.securityfocus.com/infocus/1554>, 2002.
- [3] Schneier B. Cryptanalysis of Microsoft's point-to-point tunneling protocol (PPTP)[C]. In: Proceedings of the 5th ACM Conference on Computer and Communications Security. ACM, 1998: 132–141.
- [4] Neuman B C, Ts'o T. Kerberos: an authentication service for computer networks[J]. Communications Magazine, IEEE, 1994, 32(9): 33–38.
- [5] Hellman M. A cryptanalytic time-memory trade-off[J]. IEEE Transactions on Information Theory, 1980, 26(4): 401–406.

- [6] Robling Denning D E. Cryptography and data security [M]. Addison-Wesley Longman Publishing Co., Inc., 1982: 100.
- [7] Oechslin P. Making a faster cryptanalytic time-memory trade-off[C]. In: Advances in Cryptology-CRYPTO 2003. Springer Berlin Heidelberg, 2003: 617–630.
- [8] Avoine G, Junod P, Oechslin P. Time-memory trade-offs: false alarm detection using checkpoints[C]. In: Progress in Cryptology-INDOCRYPT 2005. Springer Berlin Heidelberg, 2005: 183–196.
- [9] Narayanan A, Shmatikov V. Fast dictionary attacks on passwords using time-space tradeoff[C]. In: Proceedings of the 12th ACM Conference on Computer and Communications Security. ACM, 2005: 364–372.
- [10] Avoine G, Junod P, Oechslin P. Characterization and improvement of time-memory trade-off based on perfect tables[J]. ACM Transactions on Information and System Security (TISSEC), 2008, 11(4): 17.
- [11] Raptis A, Allison A. C. Exponential-fitting methods for the numerical solution of the Schrodinger equation[J]. Computer Physics Communications, 1978, 14(1): 1–5.

作者信息

郑中翔(1990-), 河北廊坊人, 清华大学计算机系学士, 清华大学计算机系硕士在读. 主要研究领域为密码理论与技术.
E-mail: zhengzx09@gmail.com



吉庆兵(1976-), 四川绵竹人, 硕士, 高级工程师. 主要研究领域为密码与编码方向的理论与技术研究.
E-mail: jqbdxy@163.com



于红波(1980-), 博士, 副研究员. 主要研究领域为对称密码的分析与设计.
E-mail: yuhongbo@mail.tsinghua.edu.cn