基于分布式 GPU 的彩虹表密码攻击系统①

李 聪, 叶 猛, 江 舟, 高 明

(武汉虹旭信息技术有限公司, 武汉 430074)
(光纤通信技术和网络国家重点实验室, 武汉 430074)

摘 要: 彩虹表是密码破解中最常用的工具. 利用 CPU 构造彩虹表占用时间、空间巨大, 已经成为限制彩虹表应用的最大问题. 针对彩虹表存在的查找时间、构造时间、占用内存空间瓶颈问题进行了研究. 采用高速 GPU 改进彩虹表构造方法, 建立彩虹表参数与时空关系模型, 分析得出实际中可选取的参数值. 设计实现了基于分布式 GPU 的彩虹表密码破解优化系统, 包括密文提取模块、密文分发模块、GPU 处理模块和彩虹表模块. 系统实践证明, 该方法能有效提高密码破解效率, 降低彩虹表对时间空间资源的占用, 增加彩虹表实际可破解的密码长度.

关键词: 密码攻击; 彩虹表; GPU 技术; 分布式系统

Password Cracking System Based on Rainbow Table in Distributed GPU Environment

LI Cong, YE Meng, JIANG Zhou, GAO Ming

(State Key Laboratory of Optical Communication Technologies and Networks, Wuhan 430074, China)
(Wuhan Hongxu Technologies Co. Ltd., Wuhan 430074, China)

Abstract: Rainbow table is the most commonly used tool in password cracking field. Rainbow table constructed by CPU occupies huge memory space and time, which is the biggest problem in rainbow table application. This paper concentrates on the research of the bottleneck problems of rainbow table: search time, construction time and memory space. By building a model of rainbow table parameters and time-space to analyze parameter values that can be used in application, high speed GPU was adopted to optimize construction method of rainbow table. A rainbow table password attacking system based on distributed GPU was designed and realized, including cipher extraction module, cipher distribution module, GPU managing module and rainbow table module. Test results in actual password attacking system show that the method can effectively develop password cracking efficiency, reduce rainbow table utilization of space and time and increase actual password length that rainbow table can attack.

Key words: password attacking; rainbow table; GPU technology; distributed system

随着信息数据的快速增长, 信息安全成为人们越来越关注的话题. 密码加密技术是信息安全领域中最为核心的问题之一, 密码解析技术则是密码加密的逆向过程, 将密文破译变成明文是密码解析的最终目的. 因此, 密码解析技术主要用于各类网络安全系统.

Hash(哈希)函数, 即散列函数, 将任意长度的输入映射到固定长度输出, 是一种单向密码机制, 从明文到密文的映射不可逆, 因此在信息安全领域应用广泛. 目前被广泛使用的 Hash 函数主要有 MDx 系列、SHA 系列、LM 系列等. 对 Hash 函数的解密方法主要分为暴力破解法[1]和查表法[2]. 暴力破解法即对所有可能的明文做 Hash 变换, 将所得散列值与已知散列值比较, 找到匹配的明文. 该方法需要海量运算时间. 查表法即预先对所有可能的明文做 Hash 变换并记录相应明文散列对, 破解密码时只需查找到对应散列即可. 该方法虽然执行时间比较短, 但需要海量存储空

① 基金项目:“十二五”国家科技支撑计划(2012BAH38B05)
收稿时间:2014-11-19;收到修改稿时间:2014-12-17

© 中国科学院软件研究所 http://www.c-s-a.org.cn

间. 上述两种方式在理论上都是可行的, 但实际操作时系统根本无法承受如此大的时间复杂度和空间复杂度.

2003 年, Oechslin 提出了彩虹表密码分析算法, 彩虹表对前两种方法进行时空折中[3], 将所需时间和空间控制在可接受范围内, 使得破解 Hash 函数加密密码成为现实. 但是, 由于受 CPU 计算能力限制, 彩虹表算法针对多比特数密文的分析时间和空间依旧难以满足实际需求.

2003 年之后, 众多学者开始了对彩虹表内部结构的研究, 提出了许多优化彩虹表的结构方法, 提高了彩虹表的查找效率. 近年来, 由于 GPU 性能的快速提高及其并行执行线程特性, 将 GPU 应用于彩虹表构建和查找成为了研究的热点. 截止目前, 大多数学者对彩虹表的应用研究偏向于分布式 GPU 架构上的实现和逻辑编程方法设计, 以及查找彩虹表过程中 GPU 中的线程数分析等问题. 对于分布式 GPU 构建彩虹表实现过程中的具体参数选择问题, 还没有人进行过理论研究. 因此, 本文研究量化了彩虹表构造查找过程与时间空间的关系, 建立了参数模型, 并将分析结果应用于分布式 GPU 构造、查找彩虹表过程中, 成功实现了基于分布式 GPU 的彩虹表密码解析系统. 该系统提高了密码破解效率, 克服了传统方法在时空上的局限性, 增加了彩虹表可破解的密码长度, 并为实际的密码解析系统参数选择提供了理论依据.

1　彩虹表的工作原理

1.1 彩虹表的构造原理

彩虹表的核心思想是将明文计算得到的哈希值由一个截短函数(R 函数)映射到明文空间, 从而可以交替地计算明文和哈希值[4]. 针对一个全遍历的特定样本空间, 系统随机生成很多个明文. 以 MD5 函数为例, 一条 MD5 彩虹链长 16 字节, 前 8 个字节是链头, 存放的是链起始的明文索引; 后 8 个字节是链尾, 存放的是链结束的明文索引. 每条链以链头的明文索引为开始, 得到明文 1, 在明文 1 的基础上计算链长(固定)次数的哈希和 R 函数运算, 得出明文索引 n. 由于明文索引与明文是一一对应的关系, 由索引最终可以求得明文[5]. 链长 L_c 代表每条彩虹链中节点的个数, 而不代表物理意义上的链长[6]. 因此, 一条彩虹链虽然实际只占用 16 字节存储空间, 但可表示 n 个明文. 若干个彩虹链组成一个表单, 所有的表单集合在一个文件里面, 构成彩虹表, 彩虹链形成过程如图 1 所示.

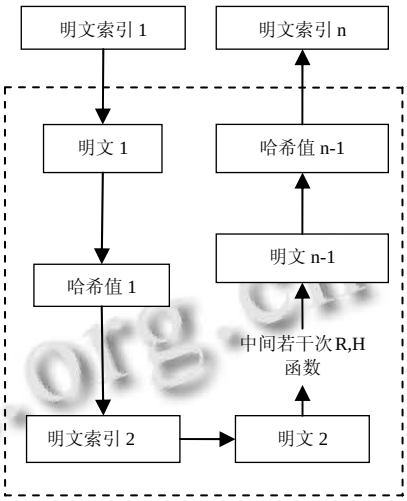

图 1　彩虹链形成示意图

1.2 彩虹表查找原理

哈希函数的加密过程为: $Q=H(P)$(H 为某种 Hash 函数, 如 MD5, Q 为加密后的密码, P 为明文). R 函数是一种特定的映射关系, 密码经过 R 函数变换后得到明文的索引, 索引对应一个明文. 每个明文都是唯一的. 彩虹表解密方法如下:

① 对 Q 进行一次 R 函数变换 $C_1=R(Q)$, 得到明文索引, 由索引得到明文 C_1, 然后将 C_1 与每条彩虹链中的 P_n(P_n 由链尾索引得到, 下文链中每个运算得到的明文用 P_x 表示, x=1, 2, ……, n, n 表示链长 L_c)对比, 如果 C_1 和 P_n 相同, 那么 P_{n-1} 很可能就是要找的明文 P;

② 从该彩虹链明文 P_1 开始, 进行 n-2 次 H、R 函数变换, 得到 P_{n-1}, 然后对 P_{n-1} 进行一次 Hash 运算, 如果得到的 Hash 值和 Q 相等, 则 P_{n-1} 就是对应的明文 P;

③ 若不相等, 则再对 C_1 进行一次 H, R 函数变换得到明文索引, 由索引得到明文 C_2, 再和每条彩虹链中的 P_n 相比, 如果一样, 那么 P_{n-2} 很可能就是要找的明文, 再通过步骤 2 来进行验证.

④ 整个过程循环直到找到对应的明文 P 为止, 若整个彩虹链循环完毕, 仍然没有找到对应明文 P, 则解密失败.

详细流程图如图 2 所示.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

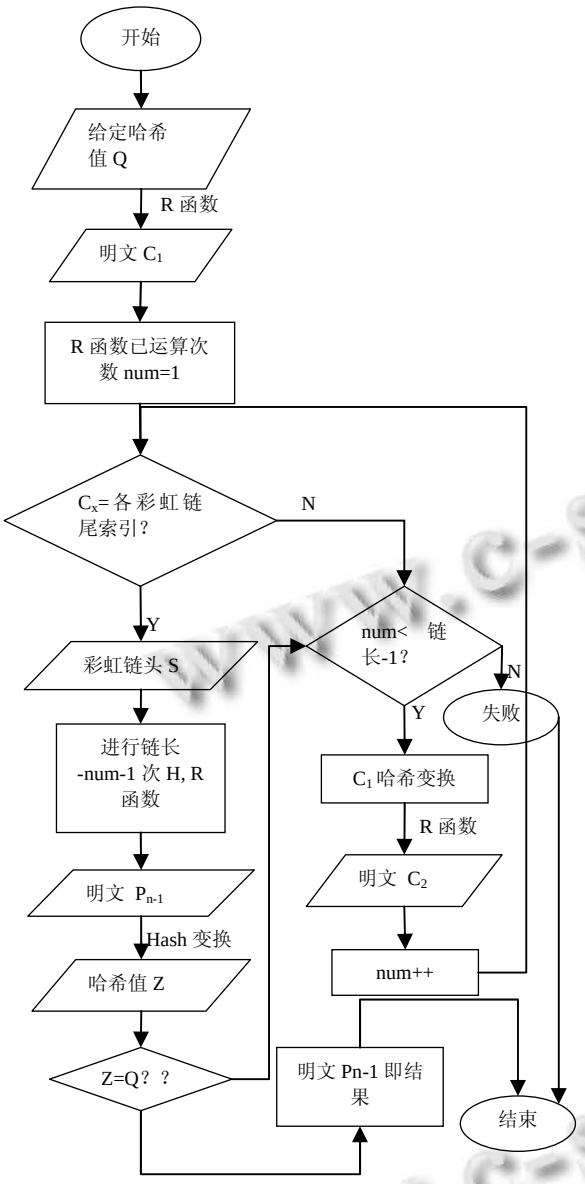

图 2 彩虹表查找流程

2 彩虹表存在的问题

2.1 空间消耗

彩虹表存在的主要问题集中在三个方面: 空间、查找时间、构造时间[7]. 以 MD5 函数为例, 在 Intel i5 CPU 上对多个密文进行实验.

实验样本空间(即明文)为小写字母+数字. 设明文可选择类型为 P, 明文长度为 $M \sim N$, 则明文组合有 $N_p = P^M+\ldots+P^N$ 种, 即样本空间数. 设生成的彩虹链长 Lc, 彩虹链数 Nc, 彩虹表单数目 Nt, 则彩虹节点数(明文数)为 $N_{nod} = L_c \times N_c \times N_t$, 必须满足 $N_{nod} > = N_p$. 由于每条链是 16 字节, 则彩虹表所占磁盘空间的大小为 $V=N_c*N_t*16Byte$, 在彩虹节点数不变的情况下, 样本空间和空间消耗成正比. 针对位数分别为 1-8 位, 1-9 位, 1-10 位, 1-11 位和 1-12 位的明文样本, 构建链长为 8000 的彩虹表, 观察所占磁盘空间. 实验结果如表 1 所示.

表 1 彩虹表所占空间与样本位数关系

样本位数	链长为 8000 的彩虹表所占空间(GB)
1-8	5.4
1-9	194.4
1-10	6998.4
1-11	251942.4
1-12	9069926.4

由表中数据可知: 样本位数每增加一位, 彩虹表空间增加约 36 倍. 当样本位数达到 11 位的时候, 构建的彩虹表达到 251942.4GB, 如此巨大的空间需求很难满足.

2.2 查找时间消耗

针对 MD5 的密文, 构造 n 条彩虹链, 链长分别为 1, 2, ……, L_n, 然后针对这 n 条彩虹链, 从链头算到链尾, 此过程平均消耗时间 $T_1= O(L_n)$. 将密文进行 R 变换, 与每条彩虹链末尾索引对应明文对比, 消耗时间 $T_2= log(Nc \times Nt)$, 此时间很小, 可以忽略. 在找到的几条彩虹链中查找具体明文, 平均消耗时间 $T_3=O(L_c)$. 因此, 总体消耗时间 $T=T_1*T_3= O(L_c)* O(L_c)$. 针对长度为 6 位的密文, 利用不同链长的彩虹表进行二分查找, 得到查找时间结果表 2 所示.

表 2 密文为 6 位的不同链长彩虹表查找时间

样本	链长	查找时间(s)
6 位	800	0.23
	2000	1.36
	5000	8.57
	6000	12.31
	7000	16.98
	8000	22.42
	40000	549.05
	60000	1244.72
	80000	2207.55

针对样本位数不同的密码采用固定链长 8000 和 80000 的彩虹表进行查找, 得出的时间结果如表 3、4 所示.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

表 3　样本位数不同的链长为 8000 的彩虹表查找时间

样本位数	链长	链数	查找时间(s)
4 位	8000	210	21.26
5 位	8000	7558	21.3
6 位	8000	272097	22.4
7 位	8000	9795520	22.98

表 4　样本位数不同的链长为 80000 的彩虹表查找时间

样本位数	链长	链数	查找时间(s)
4 位	80000	21	2064.32
5 位	80000	756	2321.25
6 位	80000	27210	2207.57
7 位	80000	979552	2306.94

由表 2~4 可知, 在不同位数下查找时间只跟链长有关. 由此可以推断 6 位 80W 链长, 跟 11 位 80W 链长的方案查找时间相同. 根据以上数据结果, 生成趋势线, 获得查找时间 y 与链长 x 关系公式.

$$y=3E\text{-}7x2.0011$$

由推导公式可知, 查找时间确实只与链长有关. 由公式算出, 11 位样本采用链长为 80W 的彩虹表查找, 时间大约为 54.14 小时, 这样巨大的查找时间完全不能满足实际需求.

2.3 构造时间消耗

通过实测链长分别为 800、8000、80000 的不同样本的彩虹表构造时间(此处数据表格太大, 没有附上), 得到对比数据, 生成趋势线, 得到单 CPU 线程下彩虹表节点个数与构造时间关系公式:

$$y=2E\text{-}7x\text{-}0.0046$$

式中, y 是构造时间, x 代表节点个数. 由上式可知, 彩虹表构造时间与彩虹表节点数成正比. 生成彩虹表, 需要遍历每一个节点, Intel i5 CPU 单线程 1 秒大概能够生成 3,103,108 个节点, 根据 2.1 节中的彩虹节点数公式, 可以大致估计出单 CPU 构建彩虹表所需要的时间, 如表 5 所示.

表 5　单 CPU 构造彩虹表所需时间

样本	节点数(=样本个数)	单 CPU 线程(天)
1-8 位数字+字母	2901713047668	11
1-9 位数字+字母	104461669716084	389
1-10 位数字+字母	3760620109779060	14025
1-11 位数字+字母	135382323952046000	504951

由表 5 可知, 单个 CPU 构造彩虹表所需时间非常久, 实际中只有 8 位以下的样本可以用彩虹表实现.

3　系统设计与实现

为了解决彩虹表存在的三个主要问题, 采用分布式 GPU 来构造和查找彩虹表[8], 并结合前述分析选取参数, 设计实现密码破解系统. 如图 3 所示, 该系统采用英伟达 GPU, 包括密文分析模块、GPU 组处理模块和彩虹表模块[9]. 密文分析模块通过模式匹配技术对加密协议数据进行预处理分析和冗余过滤, 选择性分发给 20 个 GPU, 每个 GPU 模块独立并行工作, 彩虹表模块通过 GPU 完成彩虹表初始化、构造和查找[10]. 成功解析的消息密钥会由 GPU 组处理模块自动上传到前段密文分析模块形成历史密钥[11].

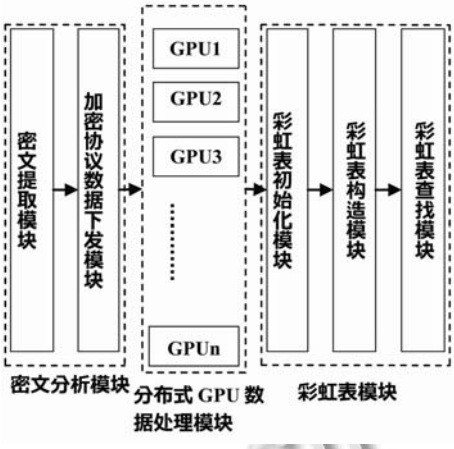

图 3　彩虹表解密系统

GPU 加速可以解决链长过长导致查找时间长的问题, 引入 20 个并行 GPU 线程, 每个 GPU 线程加速 100 倍, 总共加速 2000 倍. GPU 的使用可以解决彩虹表构造时间瓶颈和查找时间瓶颈, 要进一步解决彩虹表占用空间瓶颈, 系统参数的选取至关重要. 由上一节中分析可知, 扩大彩虹表链长来缩短彩虹表链数量, 可以大大减少彩虹表对磁盘的占用.

因此, 将彩虹表链长扩大 100 倍到 80w, 通过增加彩虹表中每条链的链长, 在总节点数不变的情况下, 彩虹链总数缩短 100 倍, 从而所占空间减小 100 倍[12]. 链长扩大前后彩虹表所占空间对比如表 6 所示.

通过表 6 数据可以观察到, 构造 11 位彩虹表只需要 2519GB 左右空间, 在实际可接受范围内.

使用该系统对常用的密码位数的数字+字母密文进行解密, 所得结果如表 7 所示.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

表 6 链长扩大 100 倍前后所占空间对比

样本	链长为 8000 所占空间(GB)	链长为 800000 所占空间(GB)
1-8 位数字+字母	5.4	0.054
1-9 位数字+字母	194.4	1.944
1-10 位数字+字母	6998.4	69.984
1-11 位数字+字母	251942.4	2519.424
1-12 位数字+字母	9069926.4	90699.264

表 7 系统性能数据

样本	破解时间(分钟)	破解成功率(%)
1-8 位数字+字母	1.6385	98.5
1-9 位数字+字母	1.67965	97.6
1-10 位数字+母	3.1587	94.2
1-11 位数字+字母	56.39	94
1-12 位数字+字母	1973.2374	89.3

上表中破解时间=彩虹表查找时间+IO 处理时间. 现商用破解系统的破解成功率一般在 80%左右, 且耗时长. 由表 7 可知, 该系统大大提升了破解时间和破解成功率, 且减少了系统占用空间. 对于常规使用最多的 1-11 位数字+字母密文, 破解时间仅需 56 分钟, 占用空间 2519GB, 成功率高达 94%. 无论在时间还是空间方面, 实际系统均可以接受.

4 结语

在实际密码解析系统中, 6~11 位密码占绝大多数, 其中 11 位密码最多. 而常规 CPU 和彩虹表的组合在密码长度超过 10 位以后在时间空间资源消耗上完全不能满足实际系统需求. 为了研究彩虹表参数对密码解析系统的影响, 本文分析了彩虹表参数与时空消耗的定量关系, 并依此设计实现了基于分布式 GPU 的密码攻击系统, 成功将密码分析时空资源需求降低了 100~2000 倍. 该系统能成功破解最长达 11 位的密码.

参考文献

1 Hellman ME. A cryptanalytic time-memory trade off. IEEE Trans. on Information Theory, 1980, 26(4): 401–406.

2 Borst J, Preneel B, Vandewalle J. On time-memory tradeoff between exhaustive key search and table precomputation. Proc. of the 19th Symposium on Information Theory in the Benelux. Veldhoven. 1998. 111–118.

3 Oechslin P. Making a faster cryptanalytic time-memory trade-off. Advances in Cryptology-CRYPTO 2003. USA. Springer-Verlag. 2003. 617–630.

4 Denning DE. Cryptography and Data Security. USA: Addison-Wesley, 1982: 66–102.

5 Avoine G, Junod P, Oechilin P. Characterization and improvement of time-memory trade-off based on perfect tables. ACM Trans. on Information and System Security, 2008, 11(4): 1–22.

6 荣凯,邱卫东,李萍.基于彩虹表的 Hash 攻击研究.信息安全与通信保密,2011(4):76–79.

7 Hong J, Jeong KC, Kwon EY, Lee IS, Ma D. Variants of the distinguished point method for cryptanalytic time memory trade-offs. Proc. of ISPEC (Internationnal Conference on Information Security Practice and Experience) 2008. Berlin. Springer Science & Business media. 2008. 151 –420.

8 Xiaoyun W, Hongbo Y. How to break MD5 and other Hash functions. Advances in Cryptology- EUROCRYPT 2005. Berlin. Springer-Verlag. 2005. 19–35.

9 金铨,谷大武,赵建杰.彩虹表密码分析算法的图形处理器优化设计与实现.上海交通大学学报,2011,45(7):1006–1010.

10 李昕,曹天杰,米国粹.基于分布式环境下的彩虹表密码攻击.计算机应用与软件,2011,28(2):290–293.

11 张庆科,杨波,王琳,朱福祥.基于 GPU 的现代并行优化算法.计算机科学,2012,39(4):304–310.

12 王紫阳,赵旺,田渝,韩远凌.彩虹表在密码破解中的应用.信息安全与通信保密,2010(11):54–58.

© 中国科学院软件研究所 http://www.c-s-a.org.cn