文章编号:1001 - 9081(2009)11 - 2939 - 03

矩阵型布鲁姆过滤器在病毒过滤防火墙中的研究

王景中,杜 飞
(北方工业大学 信息工程学院, 北京 100144)
(move170@163.com)

摘 要:针对传统的基于特征码的病毒过滤算法在实际运行中存在的效率问题,提出了一种基于矩阵型布鲁姆过滤器(MBF)的病毒过滤算法。在分析该算法的空间效率、时间效率以及错误判断率的基础上,进一步研究了它的数学模型,并给出了该算法在高速病毒过滤引擎中的设计方案。最后,通过仿真实验验证该算法的有效性和实用性。

关键词:病毒特征码;矩阵型布鲁姆过滤器;哈希算法
中图分类号: TP393.08 **文献标志码**:A

Research of matrix bloom filter in virus filtering firewall

WANG Jing-zhong, DU Fei
(*College of Information Engineering, North China University of Technology, Beijing* 100144, *China*)

Abstract: Concerning the inefficient problem of traditional signature-based virus filtering algorithm in practice, a novel virus filtering algorithm based on Matrix Bloom Filter (MBF) was proposed. Based on the analysis of the space efficiency, time efficiency and the potential effects of false positives, the mathematical model of the algorithm was studied and the design scheme of virus filters in high-speech engine was given. Finally, the simulation experimental results demonstrate the effectiveness and practicability of the proposed algorithm.

Key words: virus signature; Matrix Bloom Filter (MBF); Hash algorithm

0 引言

计算机病毒随着互联网的发展展示出了与传统单机病毒许多不同的特点[1]。由于互联网结构的无尺度特点,使得病毒的传播能以几何级数的增长传遍整个网络系统,也使计算机病毒很难被彻底清除。目前病毒发展呈现出的主要趋势是病毒与黑客程序的结合,蠕虫病毒更加泛滥,破坏性、传染性和感染性越来越大。因此,一个完善的安全体系应当是包含了从桌面到服务器、从内部用户到网络边界的全面解决方案,以防范来自黑客和病毒的威胁。基于这些需求,网关防病毒技术在目前的网络应用中变得越来越重要,人们更加关注网关防病毒技术的应用[2]。

当前主流的网关防病毒产品,其对数据的病毒检测还是以特征码匹配技术为主。病毒特征码是一个特殊的识别标记,在被同一种病毒感染的文件或计算机病毒中,总能找到这些特征码。将这些已知特征码收集起来就构成了病毒特征码数据库,这样就可以通过搜索、比较网络数据流中是否含有与特征代码数据库中匹配的特征代码,以确定被检测的数据包是否感染病毒。

随着病毒的不断增多,特征代码库也不断的增大,严重影响过滤的速度。同时一些蠕虫的特征码长度可以达到上百字节,匹配困难,耗时长。基于"部分信息判断不匹配"的原理[3],本文采用布鲁姆过滤器来进行病毒过滤,只需要哈希表 1/8 到 1/4 的大小就能解决问题,并且支持常数时间开销。将其应用于病毒过滤防火墙,对过滤的实时性将有很大的提升。

本文通过对矩阵型布鲁姆过滤器(Matrix Bloom Filter, MBF)原理的分析,提出了在防火墙病毒过滤引擎的设计方案,并通过仿真实验验证了其可行性。

1 MBF 的基本原理

1.1 布鲁姆过滤器的基本思想[4]

为了表示一个具有 n 个元素的集合 $S = \{s_1, s_2, \cdots, s_n\}$,用一个长度为 m 的位数组来描述 Bloom filter ($m > n$),初始化数组的每一个元素的值为 0。Bloom filter 使用 k 个相互独立的哈希函数 $h_1, h_2, \cdots, h_k$ 它们的值域为 $\{0, 1, \cdots, m-1\}$。为了运算的方便,假设元素经过哈希函数得到一个随机的地址,以便使一组元素的哈希地址均匀分布在整个地址空间 $\{0, 1, \cdots, m-1\}$。对于每一个元素 $s \in S$,数组中对应于 $h_1(s)$, $h_2(s), \cdots, h_k(s)$ 的元素被置成 1。查找一个元素时,如果 $h_i(s)$ 对应的位有一个不为 1,则 s 一定不在集合 S 中。如果全为 1,则以一定的错误率 p 认为 s 是集合 S 中的元素。$p = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx (1 - e^{-kn/m})^k$。其中 k 为哈希函数的个数,m 为 Bloom Filter 的长度, n 为集合 S 的大小,p 为错误率[5-6]。计算可得,当 $k = (\ln 2)(m/n)$ 时,$p_{\min} = (1/2)^k \approx 0.6185^{m/n}$。

在 Bloom Filter 中,哈希区域的每一位都被当成是独立的可寻址单元。在对集合元素进行编码时,同时使用若干个独立的哈希函数,将每一个哈希函数映射的地址都置为 1。这种编码方法摆脱了原来一个地址空间存储一个元素的方法。Bloom Filter 最大的优势是它的空间效率。与数组、Hash Table、二元树等不同,并不存储元素本身。由于 Bloom Filter 不用处理碰撞,因此无论集合元素本身有多大以及多少集合

收稿日期:2009 - 05 - 07;**修回日期**:2009 - 07 - 13。 **基金项目**:北京市教委科技面上项目(KM200910009012)。

作者简介:王景中(1962 -),男,内蒙古通辽人,教授,主要研究方向:数字图像处理与识别、计算机安全; 杜飞(1982 -),男,山西太原人,硕士研究生,主要研究方向:计算机信息安全。

元素加入到了位数组中,它在增加或查找集合元素时所用的时间完全恒定(由哈希函数的计算时间决定)。同时 Bloom 算法可能对数组中的同一个位置多次置1,这也是错误判断产生的根源[7-9]。

1.2 MBF 的基本思想

将 Bloom Filter 拓展到二维,成为 MBF[10]。MBF 的基本思想是将动态集合 S 表示成一个 $r \times m$ 的位矩阵,这个位矩阵拥有 r 行和 m 列。为了构建一个 MBF,必须首先按照应用的需要确定 r、m 和每一行的错误率 p,哈希函数的个数 k 以及动态集合的大小 n。在每个元素添加之前,需要确认这个元素应该被添加到哪一行中,所以使用第一个哈希函数 h_0 来做这项工作。h_0 是一个特殊的哈希函数,不同于在 MBF 中用到的其他哈希函数,因为 MBF 有 r 行,h_0 的哈希范围为 $\{0,1,\cdots,r-1\}$,而其他哈希函数的范围为 $\{0,1,\cdots,m-1\}$。因此,在构建 MBF 前,哈希函数的个数 k 不是真正用在 MBF 中的个数。实际上,将要使用 $k+1$ 个哈希函数,其中 h_0 是被特别定义的。假设哈希函数 h_0 是完美随机的,这样,每一行中的元素个数应该是相同的,为 n/r(每行所能容纳的最大元素个数 n_0)。查询一个元素时,对于元素 ele,首先利用哈希函数 h_0 计算这个元素可能被添加到哪一行,然后在这一行中,对于 $1 \leqslant i \leqslant k$,检查是不是所有的 $h_i(ele)$ 位都被置为1(此处本文采用逻辑与运算)。如果不是,可以确定元素 ele 不是集合 S 的一个元素;否则,以一定的错误率认为元素 ele 是集合 S 的一个元素。该算法的伪代码描述如下:

算法1 MBF_Init 初始化 MBF。

```
Input: r integer given in Rows, m integer given in Columns, A a set;
Output: M bitArray
CreatbitArray M[0,1,…,r-1][0,1,…,m-1] ←0
for i = 1 to n - 1 do
  for j =1 to k do
    M[h0(A[i])][hj(A[i])] ←1;j←j+1;
  end for
  i←i+1;
end for
Output M
```

算法2 MBF_Query 查询元素 ele 是否在 MBF 中。

```
Input: M bitArray, ele string, and ele is a key word
Output:true or false
unsigned int bFag ←1;
for i = 1 to k do
  bFag ← bFag & M[h0(ele)][hi(ele)];i←i+1;
  end for
  if bFag = = 0 then
    Output false
  else
    Output true
end if
```

2 算法分析

从空间效率、平均时间复杂度和错误率的角度来对算法进行理论分析。

2.1 空间效率

假设动态集合 S 中有 n 个元素,平均每个元素需要 L 个 ASC II 字符,若直接存于数组中,则需要 $8L(n+1)$ bit 的存储空间;若采用 MBF,仅需要 $(r \times m)$ bit 的存储空间。由 Bloom Filter 中的结论 $k = (\ln 2)(m/n_0)$ 可得 $m = (kn_0)/(\ln 2)$,在 MBF 中 $r \times m = (kn_0 r)/(\ln 2) = (kn)/(\ln 2)$,则两者所占存储空间之比为:

$$\frac{Array_{\text{Length}}}{MBF_{\text{Length}}} = \frac{8L(n+1)}{(kn)/(\ln 2)} = \frac{8(\ln 2)L(n+1)}{kn} \geqslant 5.6L/k$$

由上式可知,当 L 很大时,比值会很大,可见 MBF 的空间效率很高。

2.2 平均时间复杂度

在 MBF 中把元素的添加过程分为两步:1)使用特殊定义的哈希函数 h_0 来决定元素应该被添加到哪一行,这会花费 $O(1)$ 的时间;2)使用其余的 k 个哈希函数把元素对应的位置映射为1,这会花费 $O(k)$ 的时间。因此,添加一个元素到 MBF 中的平均时间复杂度为 $O(k)$。从 MBF 中查询一个元素的平均时间复杂度,其计算过程和添加一个元素类似,其平均时间复杂度也是 $O(k)$,是一个常数时间。

2.3 错误率

如果一个元素不是动态集合 S 的一个元素,但在 MBF 中查询时返回 true,便发生了错误。错误率 p^{MBF} 的计算如下[10]:对于 $0 \leqslant i \leqslant r-1$,MBF 的第 i 行发生错误的概率为 p_i^{BF},则 MBF 中所有的 Bloom filter 矩阵行都不发生错误的概率为 $(1-p_i^{\text{BF}})^r$。因此,至少有一个 Bloom filter 矩阵行发生错误的概率为 $1-(1-p_i^{\text{BF}})^r$,即:$p^{\text{MBF}} = 1-(1-(1-e^{-kn/mr})^k)^r$。

图1(a)中显示了当 r 与 n 变化时,错误率 p 的变化曲面,图1(b)中显示了当 $m/n = 10$,$k = 8$,r 从1到10递增时 p 的变化。当 $r = 1$ 时,MBF 即为普通的 Bloom Filter;随着 r 的增大,p 会急剧减小。图1(c)是对 p 取对数的图像,该图刻画了 p 在 r 小范围变化时快速变化的情况。可见,MBF 的错误率比普通的 Bloom Filter 要低,当数据量十分庞大时,MBF 会显示出很大的优势,同时保证了查询的常数开销。在 Bloom Filter 中可以得出结论:当 $p = 1/2$ 时,错误判断率最小,即 Bloom Filter 的位数组 m 有一半为0,另一半为1。在 MBF 中的情况类似,其位矩阵是一个只有0和1的非稀疏矩阵,且 $r \leqslant m$。如下所示:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & \cdots & 1 & 1 & 0 & 1 & \cdots & 0 & 1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & & & \vdots & \vdots & \vdots & \\ 0 & 1 & 1 & 0 & \cdots & 0 & 1 & 1 & 1 & \cdots & 0 & 0 & 0 & 1 \end{bmatrix}$$

3 MBF 在病毒过滤引擎中的应用设计

由以上的理论算法分析可知,MBF 算法的主要优点是节省存储空间和常数的查询时间以及较低的错误率。将 MBF 应用于病毒过滤引擎在速度上将有明显的优势,图2显示了其工作原理,在下一级的匹配处理中可以使用 BM 或 KMP 算法对已经匹配的关键字进行精确匹配,由于在速度上 BM 算法优于 KMP,本文采用 BM 算法。

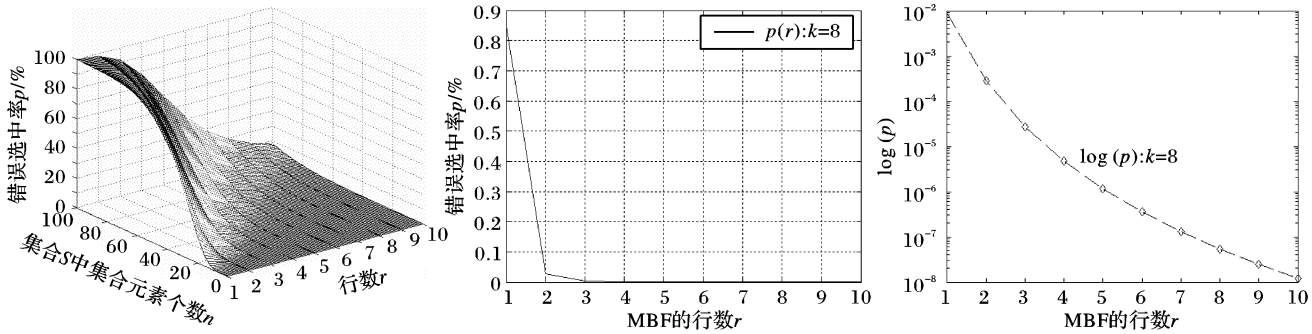
3.1 哈希函数的设计

MBF 中使用 k 个相互独立的哈希函数,常用的哈希算法有 SHA-1、MD5 和 One-way Hash 等。由文献[11]可知,在综合性能上 SHA-1 算法优于 MD5,考虑大规模应用中的执行效率,本文中采用 SHA-1 哈希算法。同一个特征码经过 k 次重复自身来计算 k 个哈希函数返回值在位矩阵中的偏移地址。

3.2 MBF 位矩阵的同步

由于动态集合 S 的可变性,在同一时刻可能出现对矩阵

的多种操作,比如一个线程正在添加元素,而另一个线程正在查询元素,为了避免出现混乱,要对 MBF 的位矩阵进行同步,对其加锁,在同一时刻只允许执行一个操作。系统的流程如图 3 所示。



(a) p与(r,n)的函数关系　(b) p与r的函数关系　(c) p与r的对数函数关系

图 1　错误率 p 与 n 和 r 的关系

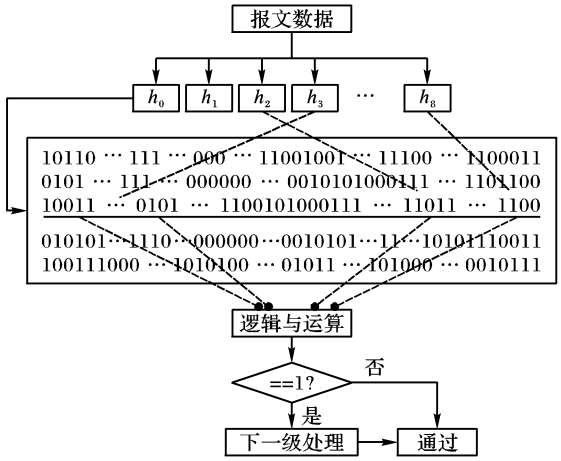


图 2　在过滤器中 MBF 的工作原理

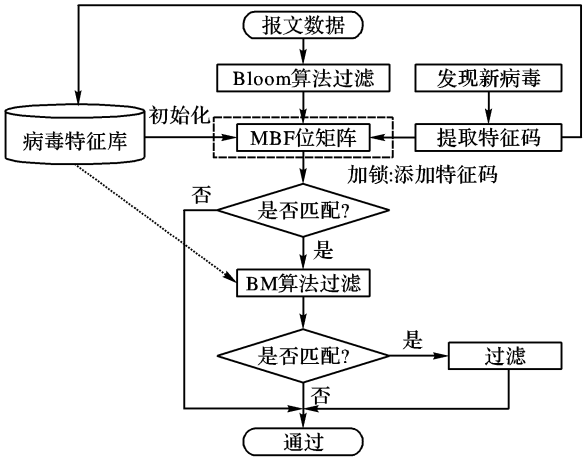


图 3　病毒过滤流程

4　仿真实验及分析

4.1　实验设置

信息流的五元素为{time,src-add,dst-add,user,data},其中,time 为时间、src-add 为源地址、dst-add 为目的地址,user 为用户,data 为信息内容。病毒过滤防火墙的病毒过滤引擎主要处理 data 部分的数据。在实验中使用 MyCCL 特征码定位器来获取病毒特征码,例如灰鸽子病毒的特征码为“00094B3E_00000003”,按照特征码的结构特点模拟 100 个病毒特征码存储于 * virCoding[100]中,MBF 的位矩阵的参数取值如表 1。

错误率 $p = 1-(1-(1-e^{-0.7})^7)^2 = 0.0163$。初始化位矩阵后其内存中的值如图 4,选择以 0x003707B8 为首地址的 16 个字节为样本,其值为“DF 45 EE C0 7F 71 18 1F 42 F8 4D 7D 82 77 FC 8C”,其中 1 的个数 63,0 的个数 65,两者近似相等,符合理论上的预期。

表 1　MBF 的结构参数

参数	值	参数	值
r	2	k	7
m	1 024	m/n	10

```
Address:  0x003707b8
003707B8  DF 45 EE C0 7F 71 18 1F 42 FB 4D 7D 82 77 FC 8C
003707C8  8A 06 A3 1C 6C 59 FF 51 E2 31 06 C2 7A 68 E1 92
003707D8  18 1E 81 B9 00 6C BA 67 95 07 84 2D EC 4D BF 65
003707E8  34 62 FA 56 92 49 68 15 E8 72 12 15 CC BD 05 A1
003707F8  D7 BC 8C 81 26 E9 03 E9 35 4D 65 5C DF 6D 40 A7
00370808  44 FC 10 15 57 A9 A9 0F 28 E5 5B 3E 5C A7 22 CA
00370818  AF D0 95 DD DC 25 CB 6C EF 9F E0 22 59 AC FC 44
00370828  46 EF BE 9D 81 E7 BA 66 13 7E B4 70 16 D3 95 9A
```

图 4　MBF 位矩阵中的值

对病毒数据的过滤采取分组实验,其中一组数据中含有真实的病毒特征码,用于检测该算法是否可以准确地过滤病毒;另一组数据不包含病毒特征码,用于检验该算法的错误判断率。

将第一组数据储于 * filterCoding[50]中,经过 MBF 过滤,可以正确地识别出病毒特征码,与瑞星杀毒软件的检测结果相同。

将第二组数据存储于 * filterCoding[300]中,模拟 data 中的数据,经过 MBF 过滤,错误选中的元素有 3 个,错误率为 $p = 3/300 = 0.01$,增加第二组的数据量,相关的实验数据如表 2。

表 2　MBF 过滤实验数据

* filterCoding[ ]	* virCoding[ ]	错误数	错误率 p (实验值)	错误率 P (理论值)
300	100	3	0.0100	0.0163
500	100	6	0.0120	0.0163
1 000	100	13	0.0130	0.0163
5 000	100	51	0.0102	0.0163

由实验数据可知,错误判断率 p 的实验值与理论值基本相符,随着数据量的增大,错误率 p 在一个稳定的范围内,可见错误率 p 由 MBF 的结构参数来决定。

4.2　实验结果及分析

1)确定一个关键字是否在集合中只与哈希函数的个数 k

(下转第 3010 页)

(a) 低分辨率图像 (b) 传统正则重建

(c) 总变分模型重建 (d) 本文重建算法

图3 超分辨率图像重建实验

3 结语

在超分辨率重建处理中,本文提出引入图像的局部空间结构特性来自适应正则化重建,采用图像的局部方差识别图像的棱边及平滑区域,针对不同的图像区域采用不同的正则化权重,该方法能有效减少超分辨率图像的重建误差。实验结果表明,该算法不仅能在重建误差上优于传统的非空间自适应正则化以及总变分模型重建算法,并且对于正则化参数的选取具有一定的鲁棒性。接下来的工作主要为根据不同的图像和噪声情况尽可能准确估计出重建过程中的正则化参数。

参考文献:

[1] TSAI R Y, HUANG T S. Multi-frame image restoration and registration [C]// Advances in Computer Vision and Image Processing. Greenwich, CT: JAI Press, 1984, 1: 317 - 339.

[2] STARK H, OSKOUI P. High resoluton image recovery from image-plane arrays, using convex projections [J]. Journal of the Optical Society of America: A, 1989, 6(11): 1715 - 1726.

[3] IRANI M, PELEG S. Improving resolution by image registration [J]. CVGIP: Graphical Models and Image Processing, 1991, 53(3): 231 - 239.

[4] SCHULTZ R R, STEVENSON R L. A bayesian approach to image expansion for improved definition [J]. IEEE Transactions on Image Processing, 1994, 3(3): 233 - 242.

[5] NHAT N, MILANFAR P, GOLUB G. A computationally efficient super-resolution image reconstruction algorithm [J]. IEEE Transactions on Image Processing, 2001, 10(4): 573 - 583.

[6] ELAD M, FEUER A. Superresolution restoration of an image sequence: Adaptive filtering approach [J]. IEEE Transactions Image Processing, 1999, 8(3): 387 - 395.

[7] BABACAN S D, MOLINA R, KATSAGGELOS A K. Total variation super resolution using a variational approach [C]// ICIP 2008: 15th IEEE International Conference on Image Processing. San Diego, CA: IEEE Press, 2008: 641 - 644.

[8] 陈远旭,罗予频,胡东成. 基于PDE正则化的超分辨率图像重构方法[J]. 计算机工程, 2007, 33(22): 4 - 5.

[9] LAGENDIJK R L, BIEMOND J, BOEKEE D E. Regularized iterative image restoration with ringing reduction [J]. IEEE Transactions on Acoustics, Speech and Signal Processing, 1988, 36(12): 1874 - 1888.

[10] MILLER K. Least squares methods for ill-posed problems with a prescribed bound [J]. SIAM Journal on Mathematical Analysis, 1970, 1(1): 52 - 74.

[11] LAGENDIJK R L, BIEMOND J. Iterative identification and restoration of image [M]. Boston, MA: Kluwer Academic Publishers, 1991.

[12] KEREN D, PELEG S, BRADA R. Image Sequence enhancement using sub-pixel displacement [C]// CVPR'88: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Ann Arbor MI, USA: IEEE Press, 1988: 742 - 746.

(上接第2941页)

有关,哈希函数的设计要考虑其地址映射的随机性和计算速度,同时 k 个哈希函数彼此相互独立,可以用多线程并行计算来处理。

2) Bloom过滤算法存在错误判断对过滤数据来说是一种“安全过度”,即病毒数据一定会被过滤,不是病毒的数据有被过滤的风险。因此在实际应用中有进一步处理的必要。

3) MBF适用于动态的集合 S 的增长,支持可扩展性;同时病毒特征码库没有删除操作,这在先天上避开了MBF的缺点。

4) MBF并没有从根本上解决基于特征码的病毒过滤技术无法预防新病毒的缺点,但它提高了过滤的实时性。

5 结语

针对病毒过滤防火墙在实际运行中存在的效率问题,本文提出了一种基于MBF存储和过滤网络数据中病毒的算法。与传统方法相比,该算法提高了病毒过滤的实时性。仿真实验的结果证明了该方法的有效性和实用性。在接下来的工作中,将研究该结构扩展到三维的情况,并考虑病毒特征码的等价划分对结构参数的影响。

参考文献:

[1] 韩筱卿,王建峰,钟玮. 计算机病毒分析与防范大全[M]. 北京:电子工业出版社,2006.

[2] 邹秋波,吴为,李之棠. Bloom filter在防火墙中的应用和研究[J]. 通信学报,2005,26(1A):158 - 162.

[3] 白建东,孙志刚. 基于Bloom Filter的报文分类算法[J]. 计算机工程,2009,35(5):108 - 110,124.

[4] BLOOM B H. Space/time trade-offs in hash coding with allowable errors [J]. Communications of the ACM, 1970, 13(7): 422 - 426.

[5] 徐克付,齐德昱,钱正平,等. 一种网络分组内容线速动态检测方法[J]. 华南理工大学学报:自然科学版,2008,36(9):15 - 19.

[6] 肖明忠,代亚非. Bloom filter及其应用综述[J]. 计算机科学,2004,31(4):180 - 184.

[7] 肖明忠,王佳聪,闵博楠. 针对动态集的矩阵型Bloom filter表示与查找[J]. 计算机应用研究,2008,25(7):2001 - 2003.

[8] DIETZFELBINGER M, PAGH R. Succinct data dtructures for retrieval and approximate membership [C]// Proceedings of the 35th International Colloquium on Automata, Languages and Programming, LNCS 5125. Berlin: Springer-Verlag, 2008: 385 - 396.

[9] CHARLES D, CHELLAPILLA K. Bloomier filters: A second look [C]// Algorithms-ESA 2008: 16th Annual European Symposium, LNCS 5193. Berlin: Springer-Verlag, 2008: 259 - 270.

[10] BRODER A, MITZENMACHER M. Network applications of bloom filters: A survey [J]. Internet Mathematics, 2004, 1(4): 485 - 509.

[11] 杨波. 现代密码学[M]. 2版. 北京:清华大学出版社,2007.