# 自然着色聚类过程中的网络安全事件计算[1]

孙美凤[1]　彭艳兵[2]　龚　俭[1]　杨　望[1]

[1](东南大学计算机科学与工程学院, 江苏省计算机网络技术重点实验室, 南京, 四牌楼 2 号, 210096)

[2](烽火通信科技有限公司, 南京, 中山南路 315 号 6 楼, 210001)

**摘要**　自然着色过程利用有部分重叠的短比特串映射，使两个哈希函数间带有相同的颜色，为判定两个 Hash 串是否同源提供了重要依据。在商集映射的视角下分析了多个不同的聚类函数间的差异和着色关系，聚类函数间的内部平衡性结合自然着色过程可以得到源串部分比特串的聚类特性。自然着色聚类过程中的 TCP 宏观平衡性仍然保持不变，利用这个特性可以从多个具有着色关系的短比特串映射的 Hash 存储空间得到如蠕虫爆发、DDoS 之类的 TCP 宏观异常中发起者、受害者的聚类信息，为网络安全检测、监测和安全事件分布评估提供了有力的支持。

**关键词**　Hash 聚类，商集映射，自然着色，TCP 宏观平衡性，安全事件分布计算

## 1　引言

文献[1]根据 TCP 交互过程提出了基于测度和测量的 TCP 宏观平衡性的概念，能够检测高速网络中的 TCP 宏观异常如 DDoS、蠕虫等。但 TCP 宏观平衡性测度只能给出 TCP 工作正常与否的结论，缺乏进一步分析和诊断所需要的流细节信息。虽然 TCP 宏观平衡性是根据 TCP 流的独立性和完整性推导出来的，是在时间维对 TCP 流报文数进行聚类后的宏观性质，但是在评估 TCP 宏观平衡性的时候，空间等其它维的信息也是必须的。因此本文将提出一种基于自然着色聚类过程的信息提取方法，希望以极少的计算资源获得更多的 TCP 流细节信息。

为了能够检测异常的受害者或者发起者，需要进一步的信息，如 IP 地址、端口等分布，可以把 TCP 流对其它的信息引入，利用流标识方法来标识聚类后的 TCP 流。一般 TCP 都是使用五元组（源/宿 IP 地址、源/宿 Port、IP 协议号）来表示的，但五元组太消耗资源，而资源消耗小的方法会丢失原始五元组的部分信息。Bloom Filter 就是一种有效压缩编号信息的手段 [2]，在路由查找[3]、分布式文件系统[4]、串匹配[5]、网络安全监测[6]等诸多领域广泛使用，在网络研究如网络抽样[7]、还原[8]、流分布估计[9]里应用广泛。Bloom Filter 使用多个短的哈希串来再现一个长的字符串所代表的空间，其具体工作原理可以参见文献[10]。从 Bloom Filter 的存储空间提取并还原原始信息面临很多困难。

Cristian Estan 等人[11]在抽样中结合使用 Bloom Filter，用于凸现样本空间的长流，短流会被略去。Cristian Estan 等人[12]使用 Bitmap，即位映象来粗略地估计流的数量。文献[13]对 Bloom Filter 的 FPR 进行分析，使用短比特串映射扩展其 Hash 函数的选择范围，但并没有给出数学上的解释。

Sketch 也是一种基于多 Hash 的数据结构，为每个 Hash 函数保持独立的 Hash 存储空间，基于一系列复杂的特殊的映射规则，其基于时间序列模型检测子流异常[14]。在 Robert Schweller 等人[15,16]论文中，提出了一种复杂的 Hash 空间维持算法，对每个流都维持一个时间序列模型。为了获得均匀的 Hash 函数，作者使用了模块化的 Hash 函数把几个短的字符串拼接成一个长的 Hash 串；在检测到异常的时候，子流空间的位置索引就可以作为逆向 Hash 的输入，而得到原始的字符串如 IP 地址，其算法复杂，不能在一个时间粒度内得到源串的信息。

利用多 Hash 关联方面的商集映射理论分析，本文用自然着色过程对 TCP 宏观平衡性进行了聚类分析，并对聚类后的 TCP 宏观平衡性进行信息提取，以获得更多关于 TCP 宏观异常流的信息。

本文的创新之处在于充分利用自然着色过程的特性，有效地提取聚类的信息，再结合 TCP 宏观平衡性，能够综合地分析和评价网络的健康状况，及时发现安全事件的详细线索。下面将这样展开本文的讨论：第 2 节将从 Hash 函数的聚类特性讨论 TCP 宏观平衡性的聚类特性；第 3 节将从多 Hash 聚类的内部平衡性的角度讨论自然着色聚类过程的相关性质，从商集映射的角度对 Hash 函数进行重新认识，给出了商集映射下着色聚类还原的数学含义；第 4 节给出了相关的实验来验证自然着色聚类还原过程中 TCP 宏观平衡性的结论和效果；第 5 节给出了结论和将来的工作。

[1] **基金项目**：本文受国家 973 计划课题（2003CB314804）、教育部科学技术重点研究项目(105084)和江苏省网络与信息安全重点实验室（BM2003201）资助。**作者简介**：孙美凤(1970-)，女，博士生，研究方向为网络安全，EMAIL：msun@njnet.edu.cn；彭艳兵（1975-），博士，研究方向为网络安全，网络行为学；龚俭(1957-)，教授，博士生导师，研究方向包括网络安全，网络行为学；杨望（1979－），男，博士生，研究方向位入侵检测系统

## 2 多 Hash 聚类的性质

哈希函数一直作为对对象（如 IP 报文）进行标识和标识简化的工具来使用。如果对象可以用一个长的字符串 $x$ 来表示的话，由于字符串 $x$ 的二进制长度较长，比如 32 比特，使得 $x$ 所构成的编号空间 $X$ 很大，而对空间 $X$ 的访问如遍历、取 Top N 等计算和空间开销比较大。同时由于对象的数量相对稀少，因而在内存中完整维护编号空间的效率是低下的。比如 IPv4 的地址空间 $X$，实际活跃的 IP 地址并不会覆盖全部的 $X$ 空间。在内存里直接维护它确需要 $2^{32}$＝4G 个内存单元，而维护 96 比特的 TCP 五元组需要更大的开销。虽然用五元组标记一个流具有精度高、运算复杂度低等优点，但大部分场合需要对 TCP 五元组进行变形，减少计算和空间开销。

常用的方法是用一个随机均匀的 Hash 函数对众多的 TCP 流作一个独立的标记，以较短小的字符串代替较长的字符串，这样 TCP 流的标识就按照 Hash 函数给定的规则分类到某个 Hash 值上，其 Hash 串就是具有某个特性流的聚类标签。利用 Hash 函数对对象进行重新编号可以提高存储和计算资源的利用效率，但要使得不同对象间冲突的可能性最小，也就是要求 Hash 函数重新编号时尽量将对象均匀分布到新编号空间，把这个空间称为 $Y$。空间 $Y$ 比空间 $X$ 的基数小几个数量级，以显著地改善资源利用效率。

把编号空间 $X$ 中的元素 $x$ 称为**源串**，编号空间 $Y$ 中的元素 $y$ 称为 **Hash 串**，Hash 函数就是重新编号的规则 $h$。于是对于源串空间 $X$ 里的元素 $x$，通过 Hash 运算映射到空间 $Y$，即 $x\in X$， $y\in Y$，有 $h(x)=y$。由于 $X$ 空间的元素数量是稀疏的，同样 Hash 运算也要求 $Y$ 空间的元素是稀疏的，附加要求 Hash 函数要尽可能把 $x$ 均匀地映射到空间 $Y$。

对于 $X$ 的一个序列 $M$，为了判断一个元素 $x'$是否属于 $M$，分两个步骤来进行：1）使用 Hash 函数 $h(*)$把 $M$ 中的所有元素映射到编号空间 $Y$ 中去，并可以按照编号进行访问。2）对于 $x'$，有 $h(x')=y'$，而如果 $y'\in Y$，则 $x'\in M$（肯定 $x'$的存在）；如果 $y'\notin Y$，则 $x'\notin M$ （否定 $x'$的存在）。步骤一被称为 **Hash 过程**，步骤二被称为 **Hash 检验**。经过这两个步骤就完成了源串 $x'$的存在性鉴别。显然用 Hash 函数作出关于 $x'$ 的否定判断是确定的，而关于 $x'$的肯定判断则与 Hash 函数的选择、$M$ 中元素的数量等有极大的关系。当使用多个独立且相异的 Hash 后，一个流可以具有多个不同的聚类标签，比较这些标签间的异同以提高 Hash 检验的精度恰好就是 Bloom Filter 这个工具所需面对的事情。

从聚类的观点来看，Hash 函数实际上是对空间 $X$ 的元素的一次划分，具有某种性质的元素 $x$ 和 $x'$经过 Hash 映射后变成了具有相同性质 $y$ 的元素，因而 Hash 函数对 $x$ 有聚类的作用。

### 2.1 Hash 的聚类特点

如果元素 $x$ 具有两个不同的属性，而这些属性与 Hash 函数独立，则 Hash 后的元素 $y$ 也应该具有聚类前源串 $x$ 的属性。对多个不同的 Hash 函数，需要多个独立的 Hash 空间 $Y^i$ 和 $Y^j$ 来保存相关属性；如果有多个不同的属性，则需要多个独立的 Bloom Filter 结构来保持这些不同的属性。属性 P 的聚类与属性的广延属性和/或强度属性有关，把它统称为属性和，使用 UP 表示，其中 U 表示属性和运算。属性 $P_1$ 和 $P_2$ 之间的关系表示为“～”。

如果 $x.P_1$ 与 $x.P_2$ 独立，且与 $h_i$、$h_j$ 和 $h_k$ 独立，对 $P_1$ 和 $P_2$ 以 $x$ 为源串分别进行 Bloom Filter 映射：

$$h_i(x.P_1)=h_i(x).P_1 = h_j(x).P_1 \qquad \text{公式 1}$$

即对于同源的比特串 $x$，不同的 Hash 映射属性不变。

$$h(x.P_1\sim x.P_2)=h(x).P_1\sim h(x).P_2 \qquad \text{公式 2}$$

即 Hash 前属性的关系等于 Hash 后的属性关系。

$$\mathrm{U}x.P_1=h_k(\mathrm{U}A_k.\ P_1)=\mathrm{U}y_k.P_1 \qquad \text{公式 3}$$

即聚类前的 $x$ 属性和的关系等于聚类后 $y_k$ 的属性和。

最简单的聚类不变属性就是对 Hash 值命中次数的 Counting 属性和源串的命中属性 Counting，所有源串的命中次数和等于聚类前商集元素的命中次数和，源串的命中次数等于其 Hash 值的命中次数。

### 2.2 聚类后 TCP 宏观平衡性的计算特性

文献[1]讨论 TCP 宏观平衡性时并没有讨论其流标识，只是对 TCP 连接在时间上进行聚类来揭示报文数量间的平衡性。本文对文献[1]讨论的 TCP 连接在与时间正交的空间维进行聚类时，聚类后的 TCP 宏观平衡性不变，可以根据前一小节的 Hash 函数的聚类特点得到。

**例 1:** 已知正常情况下 TCP 流的 SYN_ONLY 报文和与之同向的 FIN+ACK 报文在理想情况下的数量是严格配对的，则多条 TCP 的 SYN_ONLY 报文和 FIN+ACK 报文也应如此，即 SYN_ONLY . hits= (FIN+ACK). hits

对多条 TCP 流的 SYN_ONLY 报文和 FIN+ACK 报文的源 IP（SIP）进行分别的 Counting Bloom Filter 映射后，SYN_ONLY. $h_k$(SIP).hits= (FIN+ACK). $h_k$(SIP).hits，根据公式 3，原来对单个元素 $x$ 成立的属性关系，在进行聚类后，这种属性关系也成立。

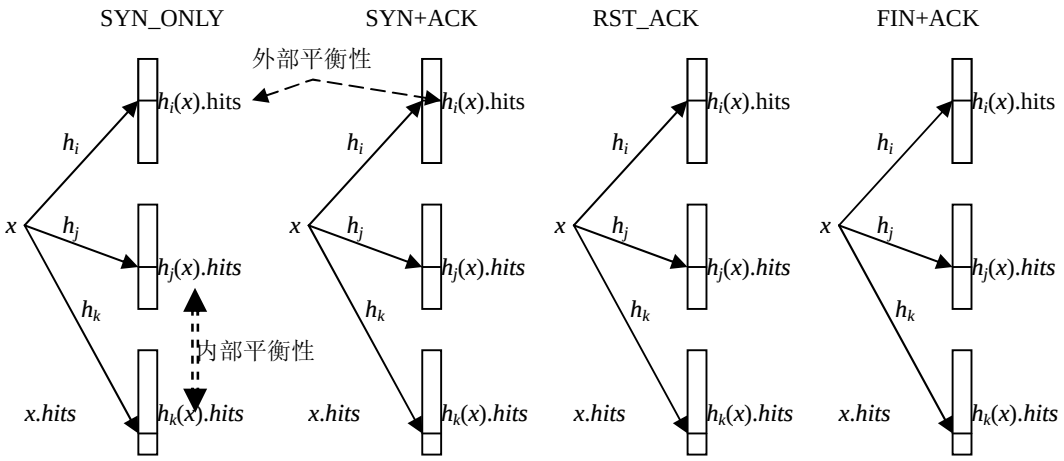



图 1 多 Hash 函数聚类后的 TCP 宏观平衡性示意图

图 1 解释了 TCP 宏观平衡性的聚类不变特性，$x$ 为流的 ID 串，如果用 TCP 五元组表示的时候一定要注意其唯一性，由于文献[1]的讨论不区分方向，可以规定某个方向上源和宿的顺序来排列五元组，来获得唯一的 ID。在合适时间粒度 $\Delta t$ 内测得各种报文的聚类 $y$ 的数量用 $y.hits$ 表示。在做 Bloom Filter 聚类的时候，给每个 TCP 报文类型使用独立的 Counting Bloom Filter 过程，如图 1 示意。

图 1 中没有列出所有的 TCP 控制报文，剩余的 TCP 控制报文的关系也与此类似。流 ID 为 $x$ 的 TCP 连接的 SYN_ONLY 的报文数可以用 SYN_ONLY.$x.hits$ 表示，其用 $h_i$ 聚类后的数量用 SYN_ONLY.$h_i(x).hits$ 表示，其他的报文和聚类函数类似。

如果流 $x$ 的数量用 TCP 宏观平衡性的测度来表示，比如 SRR＝$\Delta N_{SYN\_ONLY}$ /$\Delta N_{SYN+ACK}$，对带有流标识 $x$ 的有 SRR：

$$SRR.x = SYN\_ONLY.x.hits / SYN+ACK.x.hits \qquad \text{公式 4}$$

则根据公式 2 和公式 3，使用 Hash 函数 $h_i$ 聚类后的 SRR 为

$$SRR.h_i(x) = SYN\_ONLY.h_i(x).hits / SYN+ACK.h_i(x).hits \qquad \text{公式 5}$$

则 SRR.$h_i(x)$也可用 TCP 宏观平衡性的测度和临界值来衡量具有聚类属性 $h_i(x)$的 TCP 流是否正常。其他的 TCP 宏观平衡性测度可以类推。本文把这个属性称为 **TCP 宏观平衡性的聚类不变性**。

这意味着，如果对 TCP 流五元组进行多 Hash 聚类，其 TCP 宏观平衡性的属性将不会被改变，对非特定流的 TCP 宏观平衡性的计算可以用于检测聚类流，而检测到异常的聚类信息 $h_i(x)$就是 TCP 宏观异常中各异常报文的聚类信息。

前面公式 3 已经提到，所有源串的命中次数和等于一个 Hash 函数的所有源串的 Hash 值的命中次数和。通过这个性质可以把文献[1,2]的 TCP 宏观平衡性的测度聚类到 Bloom Filter，其 TCP 宏观平衡性保持不变。

前面讨论使用 Bloom Filter 进行聚类是在多个 Bloom Filter 结构之间进行的，其平衡性相对于 Bloom Filter 而言是外部的。下面将继续深化讨论的平衡性问题来自于 Bloom Filter 的内部，各 Hash 函数之间根据公式 1 有

**推论 1**: 定义 $H_i$ 和 $H_j$ 为 $x$ 的两个 Hash 函数，如果 $x$ 出现 $N$ 次，则 $h_i(x)$.Counting = $h_j(x)$.Counting = $N$。如果发现 $h_i(x)$.Counting>> $h_j(x)$.Counting，则 $h_i(x)$.Counting 的主成分肯定不是 $x$

这个推论可以直接通过数量关系和上面的聚类不变属性导出。对于后面一种情况，称为 $M$ 聚类于 $H_h$。这个推论讨论 Bloom Filter 各 Hash 函数间的内部平衡性，在下面的自然着色聚类过程中用来做主成分判断。

## 3 自然着色聚类过程

由于 Hash 函数是单向的，从 Bloom Filter 的 Hash 串存储空间 $A_i$ 里还原原始串 $x$ 的过程是不可逆的，而我们希

望从 TCP 五元组聚类得到的 Bloom Filter 存储空间是“可逆”的，可以还原源串的相关信息。为了找到从不可逆的 Hash 函数到可逆的 Hash 函数间的差距，下面对 Hash 函数的特点进行分析，寻找可逆和还原的可能性。

## 3.1 商集映射和“可逆”的 Hash 函数

Bloom Filter 实际上是由几个独立的 Hash 函数的特殊运算构成的一种分类方法，通过联合使用多个 Hash 函数来达到在有限的空间内降低单个 Hash 函数的甄别误差。下面从商集和商集映射的观点来重新审视 Hash 函数。

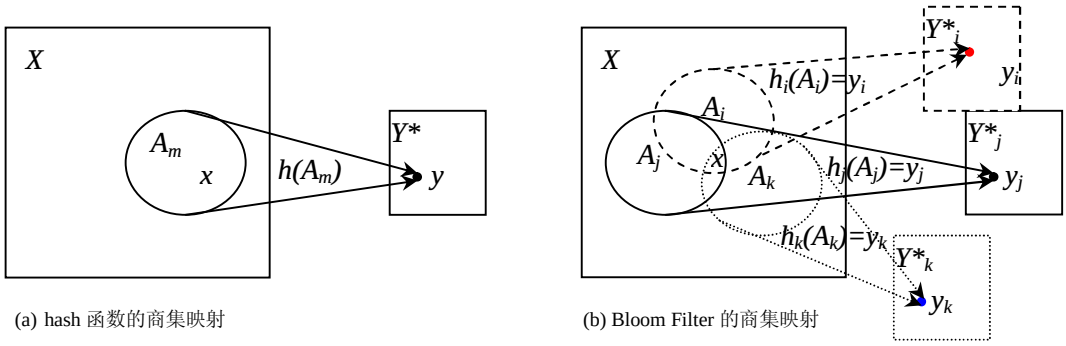
商集的定义是对 $X$ 的划分，或者说一个聚类。Hash 函数也可以看成是这样一个聚类过程。而 Hash 后元素可能对应有多个原象，可以用等价类来描述：对于每一个 $x \in X$ ，集合 $X$ 的子集$\{y \in X | xRy\}$称为 $x$ 的 R 等价类或等价类。

因此对于 $x_i$，$x_j \in A_k \in X$，$y_k \in Y$，有 $h(x_i)=y_k$，$h(x_j)=y_k$，但是 $x_i \neq x_j$，所有满足这个关系的 $x_i$ 和 $x_j$ 构成了一个等价类 $A_k$，称为 $y_k$ 的商集。而穷尽 $y_k \in Y$，可以得到所有 $A_k$，把所有的 $A_k$ 称为一个集合 $X^*$，则可以在 $X^*$和 $Y$ 间建立一个关于 $h$ 的双向单射，即：

**Hash 函数的商集映射**：$A \in X^*$，$y \in Y$，有 $h(A)=y$，$h^{-1}(y)=A$，称之为 Hash 函数 $h$ 的商集映射

使用 Hash 函数是因为下面三个理由：（1）在内存维护 $X$ 的空间不现实或者不经济；（2）$X$ 中元素构成的序列 $M$ 里的元素（其个数记为 $m$）太稀疏，在内存里维护 $M$ 的空间效率太低；（3）为了 $M$ 里少量的元素遍历空间 $X$ 的代价太大。因而如果$|X|>>|Y|>>|M|$，且冲突率较低，则使用 Hash 函数是一种替代 $X$ 维护 $M$ 的最佳方案。

上面谈到 Hash 函数对于 $x$’是否属于 $M$ 的否定判断是确定的，而对于肯定判断则因为商集映射的存在，需要进一步的讨论，因为从 $y$’推断 $x$’的时候存在着内部冲突。对于 $x_1$，$x_2 \in X$，$y \in Y$，有 $h(x_1)=y$，$h(x_2)=y$，但是 $x_1 \neq x_2$，根据上面的定义，属于同一个商集，这就是商集视角下 Hash 函数产生冲突的描述。因此需要假设 Hash 函数是均匀的，而且 $M$ 里的元素数量$|M|$较少，使得平均每一个 $y$ 对应的商集元素个数也较小，这样产生冲突的可能性也会较小。如图 2(a)和(b)所示，三个商集 $A_i$、$A_j$ 和 $A_k$ 的交集显然比原商集小很多，在$|X|>>|Y|>>|M|$时，$x$ 不仅在 $X$ 里较稀疏，其在 $A_m$ 里的均值也不到一个，即 $EXP(|A_m|) \leq 1$，因而产生 Hash 冲突的概率比单个 Hash 函数小很多。




(a) hash 函数的商集映射　　(b) Bloom Filter 的商集映射

图 2 Hash 函数和 Bloom Filter 的商集映射

Bloom Filter 使用多个 Hash 函数来进行相同的判断，扩展了 Hash 函数数量的同时也降低了其 **FPR**，下面进行分析。Hash 中冲突产生的原因是当 $x$ 映射到 $y$ 时，发现 $y$ 已被映射过了，就会产生冲突。如果用 $p_1(h,m)$表示 $m$ 个元素 $x_i \in X$（$i$=1, 2, …, $m$）被映射到空间 Y 后，Hash 函数 $h$ 在某个位置 $y$ 大于 0 的概率。显然 $p_1(h,m)$就是 Hash 函数 $h$ 的 FPR，而对于均匀 Hash，由于$|X|>>|Y|>>|M|=m$，可以认为 $p_1(h,m)=m/|Y|$。同理，对于有 $K$ 个 Hash 函数的 Bloom Filter，其 $FPR = \Pi_k p_1(h_k, m)$, $k$= 1, 2, …, $K$，这里假设各 Hash 函数使用独立的空间 $Y^*_k$ 来保存 Hash 的结果，如果 Hash 函数均匀，$FPR = \Pi_k(m/|Y_k|)$。对于使用共享空间 $Y^*$来保存所有 Hash 结果的标准 Bloom Filter 而言，$FPR = \Pi_k(km/|Y|)=K^K \Pi_k(m/|Y|)$。这个公式本质上与文献[13]的 FPR 一致。

在同取均匀 Hash 函数、相同的 $m$ 和$|Y|$的情况下，尽管独立 Hash 存储空间的开销比共享 Hash 空间的空间开销要大 $k$ 倍，但是其 FPR 却减少了 $K^K$ 倍。显然多个 Hash 函数的确扩展了单个 Hash 函数的功能。从商集映射的角度下看 Hash 函数是可逆的，只不过 $h(x)$的逆映射是一个具有聚类性质 $h$ 的集合。Bloom Filter 利用了集合里元素稀疏的性质，如果能够缩小集合的大小，将有可能通过“可逆”的 Hash 函数得到源串 $x$。

前面已经指出，Hash 函数可以看成是对 $X$ 或 $M$ 的一个聚类，聚类的规则就是 $h$，即通过 Hash 函数获得了关于 $h(A_m)=y$ 的一个商集 $A_m=h^{-1}(y)$。显然以聚类的视角来审视 Hash 函数，并不要求 $EXP(|A_m|) \leq 1$，至少可以把 $EXP(|A_m|)$ 取得大于 1。如果 Hash 函数 $h$ 选取的是使用源串某位置的部分比特串 $c$，那么 Hash 函数的语义就变成了在相同位置

上具有相同比特串 $c$ 的源串集合。

## 3.2 还原过程的定义和面临的问题

对于一些应用，已知 Bloom Filter 的 Hash 存储空间 $Y_K$，要求能够得到 $M$ 中的源串，或者得到 M 中源串的聚类特征。这个过程把它称为**还原过程**。这个过程与 Hash 过程和 Hash 检验的最大区别是，Hash 过程、Hash 检验把 Hash 存储空间当作黑盒子，而还原过程把 Hash 存储空间当作白盒子。

有三个方面的问题是源串还原中必须面对和解决的。首先，由于商集映射的关系，难以从单一的$\{y_k\}$值推导到 $x$，因为与 $y_k$ 对应的集合 $A_k$ 里的元素太多，使得这一过程实际上属于不可逆过程。其次，均匀 Hash 的一个特点是要使得 Hash 后映射的 Counting 分布均匀，Hash 函数就会选择比较复杂的计算过程。最关键的是，与 Hash 过程和 Hash 检验完全不同，还原过程不知道多个 Hash 函数值$\{y_k\}$是否来自于同一个源串 $x$，更无法确定 $x$ 的值，这使得计算过程难以展开。这个过程把它称为同源判定，是其中最为棘手的问题。

头两个问题可以通过多个 Hash 函数的语义扩展，选择适合的非均匀 Hash 函数来降低 Hash 逆过程的计算复杂度，可以带有语义，可以根据求解目标的语义环境设计出合理的 Hash 函数，而还原过程也非常简单。

为了还原源串 $x$，最简单一种的 Hash 函数是取源串 $x$ 的部分比特串进行组合，其语义相当明显，即源串的部分比特，源串的特别语义也会赋予 Hash 串相应的语义。比如 32 比特的 IPv4 地址，高 16 比特表示网络地址，低 16 比特表示主机地址，如果选择这样两个 Hash 函数，就可以得到两个聚类 Hash 过程，一个代表 16 比特的网络地址，一个代表 16 比特的主机地址。在文献[13]里把直接从源串里选择部分比特作为 Hash 函数的过程称为短比特串映射。

短比特串映射作为 Hash 函数的好处是还原过程简单，加上其语义特点，如果能够确定两个这样的 Hash 串是同源的，那么就可以通过字符串拼接来还原，至少可以拼接成一段比单个短比特串更加接近于源串长度的比特串。

因此源串还原中的头两个问题通过短比特串映射的 Hash 函数得到解决，下面重点解决 Hash 串同源性判定问题。

## 3.3 着色过程

为了确定两个 Hash 函数对应的值 $y_i$ 和 $y_j$ 对应于同一个源串，必须有一个手段来强化同源串的共同特征。下面引入的着色过程正是出于这个目的，而 Coloring($h_i$)表示 Hash 函数 $h_i$ 所具有的某种属性。

**同色关系：** 对于 $x \in X$， $y_k \in Y^*_k$， 有 $h_k(x)=y_k$，k = 0, 1, 2, ...，若存在关系 Coloring ($h_j(x)$)= Coloring( $h_i(x)$)，$j \neq i$，则称 $h_j$ 和 $h_i$ 间存在同色关系。

**着色过程：** 按照同色关系中的色值来确定两个不同 Hash 函数的 Hash 串是否同源的过程。

即两个 Hash 函数映射同一个源串 $x$ 的时候，这两个 Hash 函数还具有相同的颜色。用 $C$ 表示所有颜色的种类数量，$c$ 表示某种颜色值，可以得到 Hash 函数的商集映射下的同色关系，如图 3 所示，着色过程中具有同色关系集合的交集大小要比单纯用多 Hash 联合确定的商集的交集小很多。

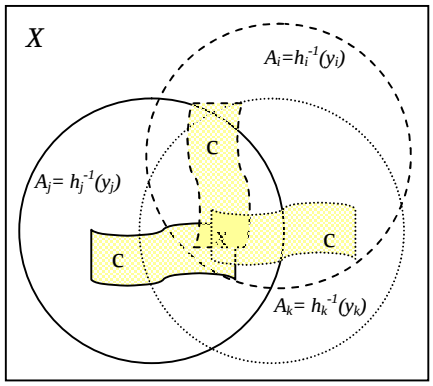



图 3 商集映射下 Hash 函数的着色过程

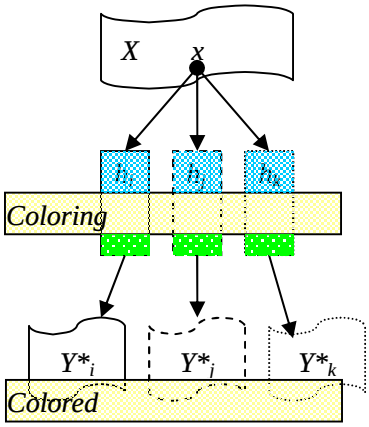



图 4 着色过程对 Hash 函数进行横向二次聚类

图 3 给出了商集映射视角下的 Bloom Filter 的同色关系图，方框代表 $X$ 的所有可能取值，$A_i$ 代表具有某种聚类属性 $h_i$ 的源串 $x$ 的集合。各种不同颜色的圆圈代表对于同一个源串 $x$ 映射到 $y_i$，不同 Hash 串 $y_i$ 所代表的源串的集合 $A_i$，则源串 $x$ 必定在三者的交集里面。各个带颜色的带状图代表不同 Hash 函数里具有某种着色的集合，显然只有不同的

带状图代表的颜色相同的时候，如图中的黄色所示，才表示这些着色集合里的元素具有同源属性，显然，只有这些带状图的交集部分才可能是同源的。同色带状图的交集显然要比圆圈的交集要小得多，表明同色关系在描述源串 $x$ 的时候有比 Bloom Filter 更低的标识冲突率。

图 4 给出了着色过程对 Hash 过程的作用过程。着色过程把 Hash 存储空间按照颜色进行了二次分类，只在相同的颜色的分类间建立关系。具有相同颜色的不同商集的交集所确定的 $x$ 的范围，显然比不使用着色过程的商集中所确定的范围要小。如果把 Hash 函数看作对 $X$ 中序列{$x$}的纵向聚类的话，着色过程是对 Hash 函数的横向聚类。这种横向聚类把 Hash 串的存储空间分成 $C$ 个类别，然后在不同 Hash 存储空间比较具有相同颜色的串间的关系，以此来确定它们是否具有同源关系了。

同色关系判断同源性的性质为：如果两个来自于不同 Hash 函数的 Hash 串 $y_i$ 和 $y_j$ 颜色值不同，可以肯定这两个 Hash 串来自于不同的源串；如果两个来自于不同 Hash 函数的 Hash 串颜色值相同，则这两个 Hash 串来自于相同的源串的可能性很大。如果 Hash 存储空间的元素数量稀疏，则可以以很高的概率肯定两个同色值的 Hash 串 $y_i$ 和 $y_j$ 来自于同一个源串 $x$。

按照着色过程的定义可知，若把两个具有同色关系的 Hash 函数的所有颜色按照色值排列成二维矩阵，只有对角线上的颜色才满足同色关系。同色关系确定同源性的时候，这些颜色值使得定位同源串的时候可以更准确，同时也有章可循，好比是拼接两幅图画的时候，可以按照两幅图画的边缘特征来进行定位拼接一般。虽然按照颜色值确定同源串的误差可以达到 $C^{-2}$。如果 $C$ 取 256(即使用 8 比特来表示颜色)，则这种精度理论上可以达到 1/65536，实际误差可能会略高。若需要更高的精度，可以使用 $n$ 元同色关系（n>2），可以达到更高的精度（$C^{-n}$），或者增大颜色数 C。

## 3.4 自然着色过程

上述定义的同色关系并没有给出具体的 Hash 函数，只给出了 Hash 函数应该具备的要求。由于本文只使用到了满足这个同色关系要求的特殊的 Hash 函数——短比特串映射，所以不再对其他类型 Hash 函数的可能形式进行描述。

当 Hash 函数选择原始元素 $x$ 的部分比特串时，如果两个 Hash 函数间存在着来自 $x$ 里相同位置的比特串部分，则把这种比特串重叠的关系称为**自然着色过程**，把使用这种 Hash 映射聚类的过程称为**自然着色聚类过程**。

可见自然着色过程使用的 Hash 函数利用了文献[13]定义的短比特串映射，自然着色过程中只需在选择两个 Hash 函数的时候有部分比特串重叠即可。而通过自然着色过程可以以较少的计算复杂度、具有直观的语义、较小的误差还原原始元素 $x$。比如用 $a$、$b$、$c$ 分别表示子串在 $x$ 里的位置，则源串 $x$ 为 $a.b.c$，两个 Hash 函数分别 $h_1=a.b$ 和 $h_2=b.c$，比较 $h_1$、$h_2$ 存储空间里语义上有相同 $b$ 的部分，在 $Y$ 中的元素较稀疏的时候，根据具有相同颜色的 b 可以唯一确定源串 $x$。如果仍然不能唯一确定 $x$，可以使用多元同色关系来提高精度。

利用自然着色过程的性质能够揭示两个具有同色关系的 Hash 串间是否具有同源属性，还原出更多源串聚类特征。

## 3.5 自然着色过程中的还原和聚类的外部平衡性

从上面的分析可以看出，自然着色由于能够确定来自于两个不同的 Hash 函数的 Hash 串是否来自于同一个源串，因此能够用于源串的还原。使用短比特串映射的 Counting Bloom Filter 使得其语义可以用来还原原始的集合 S。

如果要还原全部的串，需要把源串分成尽可能多个彼此重叠的短 Hash 串。对于重尾分布的源串如 TCP 五元组的分布，可以使用主成分分析还原消减计算的规模。

网络里主要流量聚类于少量活跃 IP，即其服从重尾分布，这样只对其 Top N 进行分析，就可以得到其主要成分的活跃情况。根据第 2 节的推论 1，可以得出更多有趣的结论。还是以源串 $x$ 表达成为的 $a.b.c$ 来作为例子，并使用自然着色过程来描述推论：

> **推论 2:** 具有相同颜色值 $b$ 的两个 Hash 函数值 $a.b$ 和 $b.c$，如果 $a.b$ 与 $b.c$ 命中率相差的比例超过一定的阈值，则源串 x 主要聚类于命中率较大的串上；反之，若比例小于一定的阈值，则 $a.b.c$ 是最有可能的聚类

这里判定“阈值”的标准比较难以给出，经验表明，一般两种比例间差 10%以上，且 Counting 不能太小，可以以 10%作为阈值。

因此在主成分分析中，Top N 的分析结果就能够给出主导成分的分布特征，特别是在源串属于 Pareto 分布的时候，计数为 Top N 里的源串里就包含了源串的主成分，此时只需要对 Top N 根据推论 2 进行相应的分析即可得到其主成分的聚类信息并计算其比例。同时由于只需要分析 Top N，计算规模也小了几个数量级。本文把利用自然同色关系和

本算法进行源串聚类信息还原的过程称为**自然着色聚类还原过程**。

这个算法得到的源串可能是短串，作用到 TCP 五元组上即是大量活跃 TCP 报文的聚类特征，也正是分析 TCP 宏观异常的时候需要的关于 IP/端口的聚类信息。根据公式 1 和公式 2，按照这个算法找到的 TCP 五元组聚类信息，其命中率仍然满足 TCP 宏观平衡性，因此根据文献[1]的测度来判断自然着色聚类还原过程得到 TCP 宏观异常的聚类信息，可以得到 TCP 宏观异常发生时的端口、网段甚至主机地址的聚类信息。这样利用自然着色聚类还原过程来提取 TCP 五元组的聚类信息，并利用 TCP 宏观平衡性的测度，就可以得到发生 TCP 宏观异常时的异常信息。

下节将给出一个实际的例子帮助理解自然着色聚类过程，并展示这个方法在网络安全领域的应用前景。

# 4 实验和分析：WIDE 数据集，蠕虫扩散

## 4.1 基本数据

本小节给出的例子采用日本的 WIDE 主干网的 trace[17]，跨太平洋的 100Mbps 链路，收集于 2005-01-07，全天 24 小时的数据，使用 tcpdpriv.[18]的一个改进版本对 IP 地址进行了净化，数据为 PCAP 格式，每 15 分钟一个文件。下面将对该 trace 进行 TCP 宏观平衡性分析，并对关键点进行自然着色聚类，以还原五元组聚类信息。

图 5 给出了这个 Trace 里的各种 TCP 报文的数量变化，横轴为 2005-01-17 当天的时间，纵轴为各类报文的数量，时间粒度为 5 分钟。从图中可以看出，SYN_ONLY 报文的数量平均要比 SYN+ACK 报文多出 10 倍左右；FIN_ONLY 报文则不到 SYN+ACK 报文的 1%。在 13:40 有一个较高的尖峰同时出现在 SYN_ONLY 报文和 RST+ACK 报文时间变化曲线上，可以看出它们之间的应答关系；而在 13:55 时刻 SYN_ONLY 发生了一次 TCP 宏观异常，可能是单纯的扫描，也可能是蠕虫爆发时的扫描。

由于 SYN_ONLY 报文平均比 SYN+ACK 报文多出 10 倍，SRR 测度（即 SYN_ONLY/SYN+ACK 报文数的比值）将会大大于 1，特别是 13:55 左右的峰值将会大于 100，如图 6 所示。图 6 给出了 SRR 的曲线，横轴是当天的时间，纵轴是 SRR 测度。图中可以看出，最高的 SRR 峰值达到了 190。

图 7 给出了其它宏观平衡性测度的曲线，这里只以图中的 B 点的 NARR 异常尖峰来对 Trace 的数据进行自然着色聚类还原分析。图中的 SRV 基本在 1 附近波动；CCR 也基本围绕 1 波动，但是幅度比 SRV 大一些。NARR 曲线主要反映 RST+ACK 报文与 SYN+ACK 报文间比例的变化，尖峰表明图 5 中 RST+ACK 的尖峰与 SYN+ACK 无关，或者说这些 RST+ACK 报文不属于完整的 TCP 流，是异常 TCP 报文到达的反映。

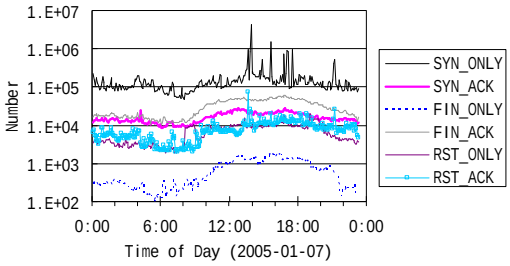



图 5 WIDE Sample B 链路上各类 TCP 报文数量

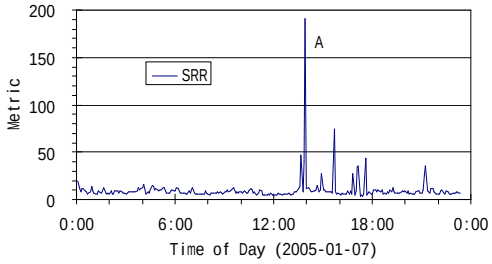



图 6 WIDE Sample B 链路上 SRR 变化曲线

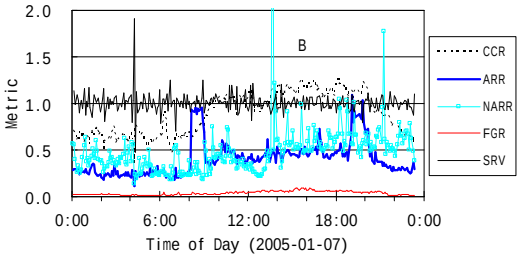



图 7 CCR、ARR、NARR、FGR 和 SRV 变化曲线

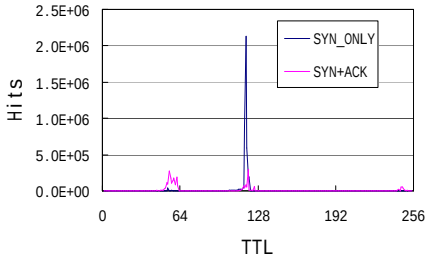



图 8 A 点蠕虫爆发的 TTL 值异常分布

本例中 SRR 长期大于 1，可以断定存在某种异常，但可以肯定不是路由循环，因为如图 8 所示，A 点 SYN_ONLY 和 SYN+ACK 的 TTL 分布曲线里没有看到路由循环典型的连续分布特征。图 8 的 TTL 曲线表明 SYN_ONLY 的起始 TTL 主要为 128，根据通常的操作系统 TTL 初始值可以发现，这些 SYN_ONLY 报文主要是由 Windows 主机发出的，

响应 RST 报文的操作系统中 Linux、Unix 和 Windows 系统都有。

## 4.2 WIDE A 点，蠕虫爆发

对于 A 点突发的 SYN_ONLY 报文的各个 Hash 函数 Top 10 的中间数据如表 1 所示，短比特串映射的 Hash 函数使用文献[13]的表示方式，$H_h$ 和 $H_m$，$H_m$ 和 $H_l$ 间均有 8 比特的重叠。可以看出活跃的前缀比较多，源前缀的集中度没有宿前缀高；宿端口高度集中在 1023，9898 和 5554 端口。表 2 的聚类分析结果也证明了这一点。

表 1 WIDE trace A 点的 SYN_ONLY 报文自然着色信息 Top 4，共 4369572 个报文

| | $IP.H_h$ | Hits | $IP.H_m$ | Hits | $IP.H_l$ | Hits | PORT | Hits |
|---|---|---|---|---|---|---|---|---|
| 源 | 213.75 | 183359 | 85.60 | 58813 | 60.225 | 58813 | 2007 | 1549 |
| | 213.85 | 158068 | 75.25 | 44857 | 102.194 | 41455 | 2571 | 1526 |
| | 213.84 | 134854 | 78.102 | 44060 | 8.113 | 33096 | 1956 | 1512 |
| | 214.140 | 122867 | 190.102 | 41455 | 25.223 | 31023 | 2090 | 1509 |
| 宿 | $IP.H_h$ | Hits | $IP.H_m$ | Hits | $IP.H_l$ | Hits | PORT | Hits |
| | 134.143 | 835649 | 143.144 | 11927 | 131.77 | 1138 | **1023** | **1414351** |
| | 134.199 | 769947 | 143.145 | 11624 | 186.14 | 845 | **9898** | **1401590** |
| | 134.235 | 739176 | 143.15 | 11012 | 156.131 | 785 | **5554** | **1392807** |
| | 134.16 | 668167 | 143.246 | 10709 | 12.199 | 768 | 445 | 70775 |

表 2 A 点 SYN_ONLY 报文的聚类信息，共 4369572 个报文

| 源 IP 前缀 | % | 源端口 | % | 宿 IP 前缀 | % | 宿端口 | % |
|---|---|---|---|---|---|---|---|
| 213.75.0.0/16 | 4.20 | | | 134.143.0.0/16 | 19.1 | 1023 | 32.4 |
| 213.84.0.0/16 | 3.62 | | | 134.199.0.0/16 | 17.6 | 9898 | 32.08 |
| 213.65.0.0/16 | 2.76 | | | 134.235.0.0/16 | 16.9 | 5554 | 31.9 |
| 214.154.0.0/16 | 2.74 | | | 134.16.0.0/16 | 15.3 | 445 | 1.62 |
| 213.88.0.0/16 | 2.66 | | | 134.196.0.0/16 | 14.0 | | |

表 3 列出的 RST+ACK 报文的聚类分析，源端口主要集中在 5554、1023、80 和 9898 端口，与表 2 的结论合并表明，RST+ACK 以 5554、1023 和 9898 对 SYN+ACK 报文进行了回应，只是规模要小多倍，根据 TCP 宏观平衡性的临界值判断方法，可以推断 5554、1023 和 9898 端口是可疑端口，而 SYN+ACK/RST+ACK 报文的 IP 前缀分布都具有典型的扫描特征。

表 3 A 点的 RST+ACK 报文的聚类信息，共 20931 个报文

| 源 IP 前缀 | % | 源端口 | % | 宿 IP 前缀 | % | 宿端口 | % |
|---|---|---|---|---|---|---|---|
| 134.199.0.0/16 | 51.9 | 5554 | 21.1 | 213.75.72.157 | 10.7 | 80 | 6.69 |
| 195.95.0.0/16 | 10.1 | 1023 | 20.7 | 212.211.148.9 | 10.5 | | |
| 87.147.204.27 | 4.4 | 80 | 13.1 | 212.214.8.157 | 10.0 | | |
| | | 9898 | 12.0 | 212.84.0.27 | 9.07 | | |
| | | | | 213.98.84.171 | 6.85 | | |
| | | | | 213.86.16.199 | 4.76 | | |

## 4.3 WIDE B 点，蠕虫扩散初期

对图 7 中 B 点的分析如下，表 4 给出了 SYN_ONLY 报文的部分聚类信息，几个 IP 主机集中了主要的但又少量的 SYN_ONLY 报文发送量(只有 1.8%－3.5%)，而宿端口主要集中在 5554、1023 和 9898 端口。表 4 给出了 RST+ACK 的部分聚类信息，源端口主要集中在上述端口，而源 IP 地址和宿 IP 地址分散。

表 4 B 点 SYN_ONLY 和 RST+ACK 报文的聚类

| | 源 IP 前缀 | % | 源端口 | % | 宿 IP 前缀 | % | 宿端口 | % |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| SYN_ONLY 925824 个报文 | 194.84.97.152 | 2.0 | | | 129.255.0.0/16 | 79.4 | **5554** | 27.1 |
| | 213.84.66.158 | 3.5 | | | | | **1023** | 27.1 |
| | 210.163.20.156 | 1.9 | | | | | **9898** | 26.1 |
| | 213.72.82.225 | 1.9 | | | | | | |
| RST+ACK 72750 个报文 | 129.255.0.0/16 | 82.7 | 1023 | 33.7 | 212.209.0.0/16 | 8.5 | 80 | 1.9 |
| | | | 5554 | 33.7 | | | | |
| | | | 9898 | 15.6 | | | | |

根据§3.5 的聚类算法，可以检测聚类端口 5554、1023 和 9898 上有 TCP 宏观异常发生，SYN_ONLY 和 RST+ACK 报文的 IP 聚类信息和端口分布可以看到明显的扫描的特征。由于此时发现了多个 IP 主机。结合 A 点的分析，A 点的 SYN_ONLY 的规模比 B 点的规模大，表明这种异常规模扩大了；回应的 RST+ACK 报文数量小了，可能此时符合回应条件的主机少了（对应于蠕虫扩散，易感主机的数量变小了）。种种迹象表明这可能是一种蠕虫的扩散行为。

查找资料发现 1023、5554 和 9898 端口是震荡波蠕虫 II（Worm.Win32.Dabber.a）的三个端口[19,20]，这三个端口都是该蠕虫传播过程中用来标识已感主机类型的；蠕虫通过向这三个端口来发送 SYN_ONLY 报文来鉴别目标主机是否存在漏洞，或是否已被感染。由于漏洞主要存在于 Windows 系统，这就解释了为什么图 8 的 TTL 的初始值都为 128。其端口和 TTL 的聚类特征还可以用于蠕虫的鉴别，用于区别蠕虫的类别和所感染主机的操作系统类型。

### 4.4 TCP 宏观异常的规模评估

由于上述三个端口为其特征端口，对应答报文对为 SYN_ONLY/RST+ACK 的端口进行统计，可以发现，B 点蠕虫扩善的初期，少数几个感染了的主机中蠕虫对外扫描的规模峰值为 743,436 次试探，存在 60,382 个易感主机对其进行响应（表 4）。而到蠕虫爆发的时候（A 点）扫描规模峰值为 4,208,748 次试探，比十五分钟前的 B 点扩大了 5.66 倍，但是其响应却只有 11,260 次（表 2、3），表明蠕虫扩散已暂时进入饱和阶段。表 2、3、4 中通过自然着色聚类还原计算得到的 IP 分布也详细揭示了蠕虫爆发的规模。这些数据可以用于蠕虫扩善的动力学模型检验。

本节的实验和分析表明，TCP 宏观平衡性在自然着色聚类过程中能够检测并计算发生异常的主机其 IP 地址和/或端口的分布信息，对于 TCP 宏观异常如 DDoS、蠕虫的检测和分布、规模的评估更加准确。

## 5 结论和将来的工作

基于 Bloom Filter 的着色过程和自然着色过程对 Bloom Filter 的 Hash 函数间进行了关联，使得可以通过这种关联来确定两个不同 Hash 函数的 Hash 值是否来自于同一个源串，从而使从 Hash 存储空间还原源串或者源串聚类特征的工作有了依据。本文解释了自然着色聚类过程的数学原理，表明其是一个全新的信息提取方法。

本文通过对商集映射的分析，得到一个“可逆”的 Bloom Filter，其利用 Hash 存储空间的稀疏性来工作；通过分析 Hash 函数的聚类特点和自然着色聚类过程，给出了一个关于从 Hash 存储空间推断和提取源串聚类特征的方法。

只要对相关的源串进行相同的聚类，TCP 宏观平衡性临界值判断在聚类后也可以进行。结合自然着色聚类过程和 TCP 宏观平衡性的结论，得到聚类串的正常/异常判定断言，并以此进行计算得到其分布规模。限于篇幅，本文仅给出了一例蠕虫爆发的检测实例，使用自然着色聚类还原方法分析得到了 TCP 五元组的聚类信息，结合聚类后的 TCP 宏观平衡性性质，能够准确地判断异常发生的类型、计算异常的发起者和受害者等更深层的细节信息，并在多种 TCP 宏观异常同在的时候表现出很好的鲁棒性。

本文的贡献在于利用自然着色过程还原主要成分的聚类信息，结合 TCP 宏观平衡性的聚类不变特性，能够检测发生在网络里的 TCP 宏观异常行为如 DDoS、蠕虫、路由循环、扫描等，并能够快速计算确定发起者/受害者的详细信息，在高速网络里检测 TCP 宏观异常行为方面和新型网管系统里有重要的意义。

由于自然着色聚类还原在有限资源状态下具有强大的信息提取功能，进一步推广其应用领域是下一步研究的重要任务。而其在网络安全里的应用，需要进行进一步的推广和部署实践。

## 参 考 文 献

[1] 龚俭, 彭艳兵, 杨望, 刘卫江. TCP 流的宏观平衡性[C]. 计算机学报. 2006 Vol.29(9) 1561-1571

[2] Bloom B. Space/time trade-offs in Hash coding with allowable errors[J]. Communications of the ACM, 1970, 13(7):422–426.

[3] Sarang D, Praveen K, David E. Longest prefix matching using bloom filters[C]. Proceedings of the 2003 conference on SIGCOMM. 2003: 201 - 212

[4] Little M, Speirs N, Shrivastava S. Using Bloom Filters to Speed-up Name Lookup in Distributed Systems[J]. The Computer Journal. 2002 Vol. 45(6): 645-652

[5] Chin-Chen C, Tian-Fu L, Jyh-Jong L. Partition search filter and its performance analysis[J]. Journal of Systems and Software. 1999, Vol: 47(1): 35-43

[6] 龚俭, 彭艳兵, 杨望, 刘卫江. 基于 Bloom Filter 的大规模异常 TCP 连接参数再现方法[C]. 软件学报. 2006, Vol. 17(3) 434-444

[7] Kumar K, Xu J, Jia W, et al. Space-code bloom filter for efficient per-flow traffic measurement[C]. INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. 2004 vol.3: 1762 - 1773

[8] Nicolas H, Darryl V. Inverting sampled traffic[C]. Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement 2003. Miami Beach, FL, USA. 2003: 222 - 233

[9] Kumar A, Sung M, Xu J, Wang J. Data Streaming Algorithms for Efficient and Accurate Estimation of Flow size Distribution[C]. in: ACM Sigmetrics 2004/IFIP WG 7.3 Performance. 2004: 177-188

[10] Andrei B, Michael M. Network Applications of Bloom Filters: A Survey[J]. Internet Math. 2003 1(4): 485–509

[11] Estan C, Varghese G. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice[J]. ACM Transactions on Computer Systems. 2003: 270 - 313

[12] Estan C, Varghese G, Fisk M. Bitmap Algorithms for Counting Active Flows on High Speed Links[C]. IMC '03. ACM Press, 2003: 153 – 166

[13] 彭艳兵, 龚俭, 刘卫江, 杨望. Bloom Filter 哈希空间的元素还原[C]. 电子学报. 2006, Vol. 34(5) 822-827

[14] Lee G, Liu H, Yoon Y, et al. Improving Sketch Reconstruction Accuracy Using Linear Least Squares Method[C]. IMC 2005. Berkeley, CA, USA, October 2005

[15] Schweller R, Gupta A, Parsons E, et al. Reversible Sketches for Efficient and Accurate Change Detection over Network Data Streams[C]. IMC 2004. ACM Press 2004: 207 – 212.

[16] Schweller R, et al., Monitoring Flow-level High-speed Data Streams with Reversible Sketches[C], In Proceedings of IEEE INFOCOM 2006

[17] http://tracer.csl.sony.co.jp/mawi/samplepoint-B/20050107/[DB/OL]

[18] ftp://tracer.csl.sony.co.jp/pub/mawi/tools/tcpd-tools.tar.gz[DB/OL]

[19] http://cert.fudan.edu.cn/articles/news.php?id=89[DB/OL]

[20] http://www.4bug.org/dispbbs.asp?BoardID=24&ID=9609&replyID=100800&skin=1[DB/OL]

# Natural Colored Information Reconstruction Process and its Application in Networking Security

Sun Meifeng[1] PENG Yanbing[2*] GONG Jian[1] YANG WANG[1]

([1.]School of Computer Science and Engineering, Southeast University, Key Lab of Networking of Jiangsu Province, Nanjing, Sipailou, 210096)

([2.]FIBERHOME TELECOMMUNICATION TECH CO.LTD, Nanjing, Zhongshan South Road, 315#, 6F, 210001)

**Abstract** The natural coloring process deploys the overlapped Short Bit String Mappings to keep the coloring relationship among different Hash functions in affinal Hash string seeking. The aggregation relationship and coloring relationship is analyzed from the view point of Quotient set mapping, it suggests that the natural coloring process can disclose more bits of original string in the multi Hash aggregation by inner balance of Hash functions. The quantitive balance of TCP packets in the natural coloring aggregation process can be used to detect the IP address(es) and/or port information of victims and/or attackers of TCP macroscopical abnormal behavior such as DDoS, internet worm etc., which was validated by the results of two experiments in the real traces. The natural colored aggregation process extends the application field of quantitive balance of TCP packets greatly in networking security detecting /monitoring fields.

**Keywords** Hash Aggregation, Quotient Set Mapping, Natural coloring, Macroscopical TCP Quantitive Balance, Security Event Distribution Computing