第 38 卷 第 3 期
Vol.38 No.3

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

• 安全技术 •

文章编号：1000—3428(2012)03—0130—04 文献标识码：A 中图分类号：TP393

基于伪名的 VANET 恶意节点检测研究

张智勇，马建庆，张世永

(复旦大学网络与信息安全研究所，上海 200433)

摘 要：针对车载自组网(VANET)节点通信时间短、实时性要求高的特点，设计一种压缩型 Bloom Filter 机制，并将其应用于基于伪名的 VANET 恶意节点检测中。该 Bloom Filter 机制能减少伪名恶意节点集合的数据存储量，以及节点数据更新时的信息交换量，同时获得更高的恶意节点检测率和更低的假阳性率。分析结果表明，该方法可降低 VANET 节点间实际传输的数据量和通信开销，提高通信实时性。

关键词：车载自组网；安全性；恶意节点检测；伪名；Bloom Filter 机制

Research on Pseudonym-based Malicious Node Detection in VANET

ZHANG Zhi-yong, MA Jian-qing, ZHANG Shi-yong

(Network and Information Security Laboratory, Fudan University, Shanghai 200433, China)

【Abstract】To meet the requirement of short communication time and real-time in Vehicular Ad Hoc Network(VANET), this paper designs a mechanism using compressed Bloom Filter mechanism, and realizes pseudonym-based malicious node detection in VANET. The mechanism can effectively reduce the data size of pseudonym set of malicious nodes and the exchanging information content in data updating process with lower false positive. Analysis results show that this method can reduce data transmission and correspondence expenses, and improve real-time of communication.

【Key words】Vehicular Ad Hoc Network(VANET); security; malicious node detection; pseudonym; Bloom Filter mechanism

DOI: 10.3969/j.issn.1000-3428.2012.03.044

1 概述

近年来，随着 Ad Hoc 网络的发展和人们对智能交通的的日益关注，出现了新型的 Ad Hoc 网络应用——车载自组网(Vehicular Ad Hoc Network, VANET)。VANET 的部署为车辆提供了相互通信的能力，在事故预警、保障交通安全以及为用户提供舒适的驾驶环境等方面起到了巨大作用[1]。相比普通 Ad Hoc 网络，一定路径(公路)形成的线性结构，节点数量多、存储空间有限、计算能力强、运动速度快、拓扑结构变化大等，都成为 VANET 的显著特点。其中，隐私性是 VANET 的基本安全需求，伪名机制[2-4]的引入使移动车辆节点通过不断更换伪名降低被跟踪或定位的潜在威胁。到目前为止，伪名机制是比较适合保护 VANET 隐私的一种有效手段。

同时，VANET 的安全问题也得到越来越多的关注。恶意节点可以利用节点间的相互通信对其他节点进行攻击和欺骗，比如通过向网络中发布虚假位置信息扰乱路由，通过篡改报文信息或插入虚假报文进行基于位置的攻击，造成车辆定位失效；车辆节点可能因为自私，欲占用某道路，发送该道路拥塞的虚假报文，从而诱骗其他车辆节点选择另外的道路[5]。更严重的是，伪名机制的引入使恶意车辆节点能通过不断改变伪名来从事各类攻击、欺诈或逃避事故责任。因此，车辆节点需要对与自己进行通信的节点进行检测，判别可能存在的恶意节点行为。

VANET 的自身特点要求在检测过程中减少伪名集合存储的数据量、减少节点每次更新数据时的信息交换量，即要求所传输的数据报长度尽可能短。考虑其很强的时间敏感性，要求检测速度尽可能快，同时这种检测是“宽容”的，允许一定错误率的存在。

本文基于 VANET 存在网络基础设施的假设实现恶意节点检测。当路边基础设施存在时，通过路边基础设施向认证中心报告其他车辆的通信报文和行为的一致性；若路边设施暂时缺失，则延迟至遇到基础设施时向认证中心报告。同时，通过路边基础设施进行车辆节点自身数据库的更新(其他车辆节点的可信度更新)，以便在网络基础设施不可用时，车辆自主检测恶意节点和判断报文的恶意性。

本文在基于伪名的 VANET 恶意节点检测过程中应用 Bloom Filter 机制的应用模型，介绍 Bloom Filter 的概念并设计在检测过程中用到的 Bloom Filter 算法 BF_Detection，同时分析该算法的性能，从而讨论 Bloom Filter 的变种——压缩型 Bloom Filter 在通信开销以及假阳性率方面的性能。

2 恶意节点检测模型

将道路上的真实情况进行抽象，得到如图 1 所示的恶意节点检测的通信过程。在图 1 中，对恶意节点检测的通信过程进行抽象，当节点 $Node_1$ 作为恶意节点，攻击或欺骗节点 $Node_2$ 时，$Node_1$ 首先在本地产生伪名以隐藏身份，同时发送消息欺骗 $Node_2$，$Node_2$ 接到消息后，从消息中得到 $Node_1$ 用

基金项目：国家自然科学基金资助项目“车用 Ad Hoc 网络的隐私与安全技术研究”(60803117)

作者简介：张智勇(1984－)，男，硕士研究生，主研方向：车载自组网，信息安全；马建庆，讲师；张世永，教授

收稿日期：2011-05-18 **E-mail：**zhangzhiyong@fudan.edu.cn

来发送消息的伪名，利用 Bloom Filter 检测该伪名是否属于本地已有的恶意节点伪名库[6]，若属于，则直接舍弃数据包；反之，判定为正常节点发送的消息，转发收到的消息。

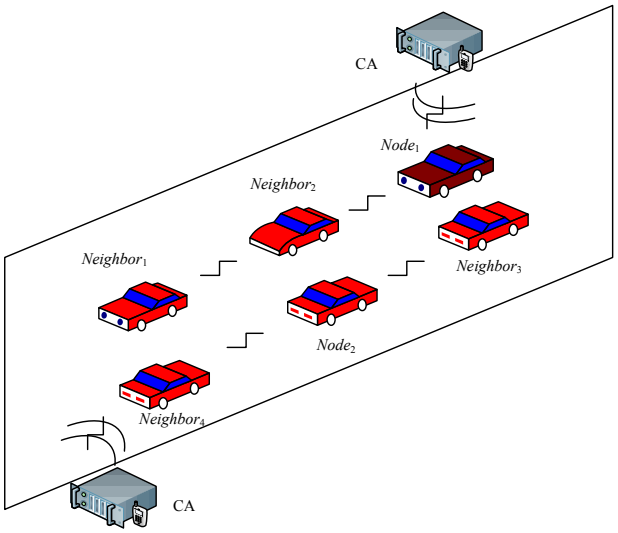

图 1　恶意节点检测的通信过程

当认证中心(CA)收到 $Node_2$ 转发的消息后，对消息进行验证，更新自身恶意节点伪名库，并每间隔一定时间广播最新的伪名库，恶意节点检测模型如图 2 所示。

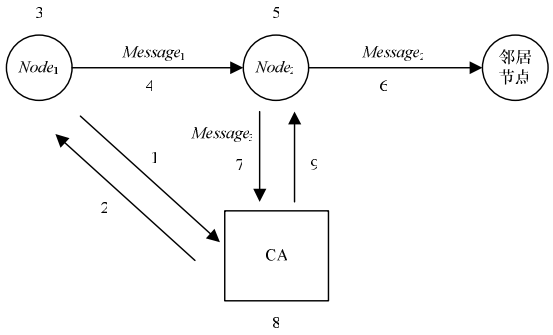

图 2　恶意节点检测模型

假设 $Node_1$ 为恶意节点，利用自己的身份标识在本地通过伪名机制产生伪名，伪名可以为多个。假定 $Node_2$ 为正常节点，同时 $Node_2$ 周围存在邻居节点 *NeighborNode*，所有节点相互间可以通信，同时设定 CA 为颁发证书、发布恶意节点伪名库的权威机构。

恶意节点检测模型的过程描述如下：

过程 1 $Node_1$ 向 CA 申请证书，节点根据自己唯一的身份标识 ID 向 CA 申请数字证书。

过程 2 CA 向 $Node_1$ 发放证书，CA 向节点发放数字证书的过程，节点以该证书作为产生伪名和在网络中证明自己具有通信权限的标识。

过程 3 产生伪名，$Node_1$ 在本地通过伪名机制产生伪名，Node1 可以依据自己的通信状况进行伪名更新。

本机制根据文献[6]中的伪名生成方法，要求节点在伪名计算的过程中应用获得的合法证书作为加密和签名的公钥，同时，节点可自主为伪名生成私钥。

过程 4 发送消息，$Node_1$ 随机选定过程 3 产生的伪名用来攻击 $Node_2$ 的伪名(假定它为 $Node_1\_Pseudonymx$)，向 $Node_2$ 发送消息 $Message_1$，考虑 VANET 对数据传输的开销和时间延迟十分敏感，因此，本文设计 $Message_1$ 的发送结构为二元组：$Msg<Node_1\_Pseudonymx, msgContent>$，数据包的部分格式如下：

…	$Node_1\_Pseudonymx$	*msgContent*	…

过程 5 检测判断恶意节点，$Node_2$ 收到 $Node_1$ 的消息后，对 $Node_1$ 进行检测，判断其是否为恶意节点。

过程 6 转发正常节点消息，$Node_2$ 完成恶意节点检测过程，直接丢弃恶意节点发来的消息，将正常节点发来的消息通过自己的伪名向邻居节点 *NeighborNode* 转发。

过程 7 向 CA 报告，$Node_2$ 完成在本地进行的恶意节点检测后向认证中心 CA 报告。其中，$Message_3$ 的发送结构为三元组：$Msg<Node_2\_Pseudonymy, Node_1\_Pseudonymx, msgType>$，$Node_2\_Pseudonymy$ 为 $Node_2$ 发送消息时使用的伪名，*msgType* 为 boolean 类型，供发送节点告知 CA，消息来自正常节点(置为 1)，或来自恶意节点(置为 0)。数据包的部分格式如下：

…	$Node_2\_Pseudonym_y$	$Node_1\_Pseudonym_x$	*msgType*	…

过程 8 CA 消息验证，认证中心 CA 对 $Node_2$ 反馈的消息进行验证，分析得到伪名对应的 $Node_1$ 的证书，根据本地维护的证书信息，识别伪名对应的 $Node_1$ 的真实身份，对 $Node_1$ 的信用值进行更新，并与节点 ID 的信用值阈值 M 进行比较，若 ID 的信用值低于 M，则判定 ID 为恶意节点，其关联的伪名皆为恶意节点，将所有伪名加入到恶意节点伪名集合，对本地维护的恶意伪名库进行更新。

过程 9 广播更新伪名库，认证中心 CA 更新恶意节点伪名库后，对 $Node_2$ 的本地恶意节点伪名集合进行更新，即向 $Node_2$ 发送更新后的伪名集合数据，并由 $Node_2$ 在本地将原有集合与接收到的集合数据做“并”运算。

3　Bloom Filter 在 VANET 恶意节点检测中的应用

Bloom Filter 是一种空间效率很高的随机数据结构，它利用位数组很简洁地表示一个集合，并能判断一个元素是否属于这个集合，被广泛应用到各种计算机系统[7-8]中表达庞大数据集及提高查找效率。

但是，Bloom Filter 的高效有一定代价：在判断一个元素是否属于某个集合时，有可能会把不属于这个集合的元素误认为属于这个集合。在能容忍低错误率的应用场合下，Bloom Filter 通过极少的错误换取存储空间，考虑到 VANET 对减少在检测过程中伪名集合存储的数据量、减少节点每次更新数据时信息交换量的要求，因此，引入 Bloom Filter 是可行的。

下面结合 Bloom Filter 在检测中的应用阐述过程 5 检测判断恶意节点的具体过程。

3.1　恶意节点的检测过程

恶意节点检测过程分为以下步骤：

步骤 1 $Node_2$ 接到 $Node_1$ 的消息后，对数据包进行分解，获得 $Node_1$ 用来发送本次消息的伪名 $Node_1\_Pseudonymx$。

步骤 2 $Node_2$ 根据获取的伪名，对其进行 Hash 过程，利用 Bloom Filter 判断伪名是否在本地已有的恶意节点伪名集合中。

步骤 3 $Node_2$ 根据检测结果对接收的消息进行操作，若不在恶意节点伪名集合中，则转发消息给邻居节点；若在恶意节点伪名集合中，则丢弃所有该伪名发来的数据包，并将此伪名隔离。最后，$Node_2$ 向认证中心 CA 反馈消息，以便 CA 中心维护车辆黑名单数据库。

3.2　Bloom Filter 的 VANET 应用设计

Bloom Filter 通过一组 k 个相互独立的定义在 n 个输入元

素上的哈希函数，将上述 n 个元素映射到 m 位的位串 V 上，它提供插入、查找等操作。

根据上述过程，设计应用 Bloom Filter 的 BF_Detection 算法，具体描述如下：

(1)基础设施已检测到的恶意车辆节点的伪名集合 *M_Node* 的初始化过程：

```
BitString BF_mNode (Set M_Node, int m){
//采用位长 m 的位串 V 表示数据集合 M_Node
BitString V[m]=0;
for(j=1;j<n+1;j++)
for(i=q;i<k+1;i++)
V[hi(M_Node[j])]=1;
 return V;
}
```

在实际应用过程中，Bloom Filter 对恶意节点伪名集合的表示过程集中发生在 CA 处，而在节点本地的初始化过程中只需将位串每一位置为 0，并通过算法第(3)步进行相应的位串操作即可。

(2)判断节点伪名是否为恶意的算法：

```
Boolean isMaliciousNode(BitString V, String Psy){
//返回(1)，if Psy∈M_Node，反之，返回 0
int i=1;
while((V[hi++(Psy)]==1)&&(i<k+1))
 if (i≤k) return 0;
 else return 1;
}
```

(3)恶意节点伪名集合元素增加时 V 的更新算法：

```
//对于在 CA 处验证后更新的数据，只要取 2 个集合的并就可
//实现操作
BitString M_Node_Union (BitString V, BitString V_Update){
 return V=(V | V_Update);
}
```

在此举一个例子对上面的算法加以说明：为方便起见，令 k=2、m=8、n=2，不妨设恶意伪名集合为 M_Node= {$Node_1$_$Pseudonym_1$, $Node_1$_$Pseudonym_2$}，哈希函数分别为 $hash_1$、$hash_2$。

在初始状态，位串 V 各位都为 0：

0	0	0	0	0	0	0	0

设 $hash_1(Node_1\_Pseudonym_1)=3$、$hash_1(Node_1\_Pseudonym_2)=7$、$hash_2(Node_1\_Pseudonym_1)=4$、$hash_2(Node_1\_Pseudonym_2)=2$，经过初始化后，位串 V 变为：

0	0	1	1	1	0	0	1

若需要检验的伪名为 $Node_1\_Pseudonym_3$，则对其分别用 $hash_1$、$hash_2$ 进行计算，若命中位串中的 1，则判断 $Node_3$ 在集合中，但是若 $hash_1(Node_1\_Pseudonym_3)=2$ 且 $hash_2(Node_1\_Pseudonym_3)=7$，则 $Node_1\_Pseudonym_3$ 同样被判断为属于集合元素，显然这样的判断是错的，这种情形即为假阳性现象，即原本不属于集合的元素被误判为属于集合。

若有 2 个集合，对应的 Bloom Filter 产生的位串 V_1 为：

0	0	1	1	1	0	0	1

对应的 Bloom Filter 产生的位串 V_2 为：

0	1	0	0	1	1	0	1

经过步骤 3 的更新后，得到新的位串 V 为：

0	1	1	1	1	0	0	1

假定恶意节点的每个伪名约为 50 个 ASCII 码字符，则直接存储这个集合需要 400(n+1) bit 的存储空间，在应用 Bloom Filter 后仅需要 m bit 的存储空间，而在实际中 $m \ll 400n$。

以图 2 中的过程解释恶意节点的产生以及车辆节点对恶意节点的检测判定过程。仍然假设节点 $Node_1$ 作为恶意节点，攻击或欺骗节点 $Node_2$，且 $Node_2$ 已通过 *BF_mNode* 以及 *M_Node_Union* 在本地完成伪名集合(设为 V)初始化和更新。

假设 $Node_1$ 对 $Node_2$ 发起攻击时使用的伪名为 *psy*。$Node_2$ 在收到该伪名发来的消息后，计算 *isMaliciousNode*(V, *psy*)，若结果为 1，则判定伪名 *psy* 为恶意节点，同时舍弃所有接收的来自 *psy* 的数据包；若结果为 0，则判定伪名 *psy* 为正常节点，对数据包予以转发。检测过程结束后，$Node_2$ 依照图 2 中过程 7 向 CA 报告。

当结果为 0 时，存在伪名为恶意节点但是 $Node_2$ 本地恶意节点伪名集合中未包含该伪名的情况。在 $Node_2$ 依照图 2 中过程 7 向 CA 报告后，通过图 2 中过程 8 的 CA 消息验证和过程 9 的广播更新伪名库使 $Node_2$ 本地恶意节点集合得到更新，假设广播的伪名集合为 V_Update，则 $Node_2$ 接收后计算 *M_Node_Union*(V, V_Update)得到最新的恶意节点伪名集合。

4 基于伪名的 VANET 恶意节点检测方法

4.1 假阴性率的分析

假阴性现象即在检测过程中，将本属于恶意伪名集合的伪名判断为不属于集合，因此，造成恶意节点判别错误的情况。假阴性率即这种情况出现的概率。

在本文设计的基于伪名的 VANET 恶意节点检测机制中，Bloom Filter 的特点决定了只要伪名是所查询集合中的元素，则方法 *isMaliciousNode*()得到的结果必然为 1，即不存在假阴性的情况发生，因此，该检测机制中的假阴性率为 0。

4.2 假阳性率的分析与证明

假阳性现象即在检测过程中，将本不属于恶意伪名集合的伪名判断为属于集合，因此，造成正常节点判别错误的情况。假阳性率即这种情况出现的概率。分析 BF_Detection 算法可知：(1)查找算法时间复杂度为 $O(k)$；(2)实现 2 个集合并的 Bloom Filter 向量表示的时间复杂度为 $O(m)$；(3)在算法中如果某一位多次被置为 1，那么只有第 1 次的操作有效，导致产生假阳性现象，因此，假阳性率成为 Bloom Filter 应用的关键点。

4.2.1 Bloom Filter 的假阳性率

根据前述设计，在初始状态时，m 位的位串 V 每一位都置为 0，通过位串对恶意伪名集合 *M_Node* 进行表达，在这个过程中，Bloom Filter 使用 k 个独立的哈希函数，分别将集合中的每个元素映射到{1,2,…,m}的范围，对任意一个元素 *M_Node*[j]，第 i 个哈希函数映射的位置 h_i(*M_Node*[j])被置为 1，即 $V[h_i(M\_Node[j])]=1$，其中，$1 \leqslant i \leqslant k$。

令 p_{err} 为假阳性率，同时为了简化模型，假设 $kn < m$ 且各个哈希函数完全随机。当集合 *M_Node* 的元素在被 k 个哈希函数映射到 m 位的位串中时，每次赋值后使某位为 1 的概率为 1/m，则为 0 的概率为(1–1/m)，把 *M_Node* 完全映射到位串中，需要做 kn 次哈希。因此，算法结束后，位串中某一位还是 0 的概率 p_0 为：

$$p_0 = (1-\frac{1}{m})^{kn} \approx \mathrm{e}^{-kn/m}$$

令 $p=\mathrm{e}^{-kn/m}$，并且令 ρ 为位串中 0 的比例，ρ 的期望 $E(\rho)=p_0$，则在 ρ 已知的情况下，要求的假阳性率为：

$$p_{\mathrm{err}}=(1-\rho)^k\approx(1-p_0)^k\approx(1-p)^k$$

考虑涉及的近似计算，p_0 只是 ρ 的数学期望，因此，ρ 的实际值可能偏离它的期望。文献[9]已经证明，位数组中 0 的比例非常集中地分布在它的数学期望值的附近，因此，近似是成立的。

将求得的 p 和 p_0 代入式 p_{err}，得到：

$$p_{\mathrm{err}}^{'}=(1-(1-\frac{1}{m})^{kn})^k=(1-p_0)^k$$

$$p_{\mathrm{err}}=(1-\mathrm{e}^{-kn/m})^k=(1-p)^k$$

给定 m、n，可求 p_{err} 的最小值，易知当 $k=(m/n)\ln 2$ 时，p_{err} 可取得最小值 $p_{\mathrm{err\_min}}$：

$$p_{\mathrm{err\_min}}=(1-\frac{1}{2})^k=(\frac{1}{2})^{(m/n)\ln 2}=0.618\ 5^{m/n}$$

即 $k=(m/n)\ln 2$ 是 Bloom Filter 取得最小假阳性率的最优哈希函数的个数。

然而，在实际应用中，通常 k 较小且固定，因此，需要根据实际情况在 p_{err} 和 m 之间作权衡。通过图 3 可以看到在 k 值固定情况下，如何在两者间权衡，同时也可以看出：k=8,16,32 时，曲线比较接近，使用 8 个哈希函数与使用 32 个哈希函数接近，同时节约近 3/4 的哈希计算时间。

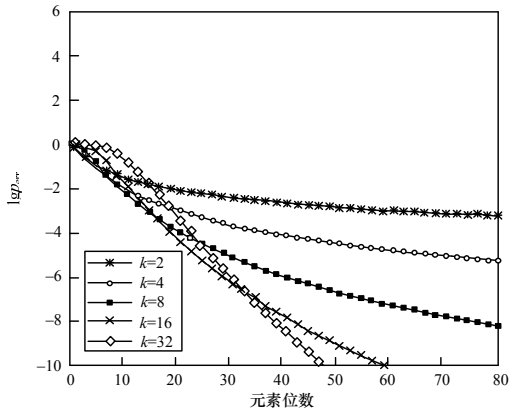

图 3　在不同 k 值下 p_{err} 值随元素位数的变化

由图 3 可以看出，同样的 p_{err} 值，不是 k 值越大越好；在相同 k 值的情况下，要获得较好的假阳性率，需要较大的 m。但是，m 的增大会对网络的通信开销产生很大的影响。综上所述，固定元素位数(m/n)，增大 k，可以获得更好的假阳性率。

在时间应用中，VANET 对检测速度的需求要求计算量尽可能小，而对于通信开销的限制要求传输位数尽可能少，因此，取 k=8、m/n=60 时，p_{err} 值接近于 10^{-7}。这并不是最好的选择，下文将对现有的 Bloom Filter 进行改进。

4.2.2　压缩型 Bloom Filter 的性能改进

在 VANET 应用中要求尽可能降低通信开销，引入压缩技术，即使用压缩型 Bloom Filter[9]减少信息量的交换。

由上面的分析可知，当 $p_0=\frac{1}{2}$ 时，即 $k=(m/n)\ln 2$ 时，得到最小的假阳性率 $p_{\mathrm{err\_min}}$，假定传输时的实际位长为 z，则有 $z\leqslant m$。下面讨论 z 与 m 的关系。

由香农编码原理[10]可知，设可能传输的符号集合为字母表 Σ。符号 s 的编码位数，称为信息量，记为 $I(s)$，则有 $I(s)=-\mathrm{lb}pr(s)$，其中，$pr(s)$ 为符号 s 的出现概率。整个字母表中的每个符号的平均信息量称为熵，记为 H，有：

$$H=\sum_i pr(s)\cdot I(s)=\sum_i -pr(s)\cdot\mathrm{lb}pr(s)$$

在本文应用中，字母表 $\Sigma=\{0,1\}$ 会因 0 出现的概率 p_0 的变化而受到影响，则记：

$$H(p_0)=-p_0\mathrm{lb}p_0-(1-p_0)\mathrm{lb}(1-P_0)$$

在这个熵值的情况下，对 m 的位串进行压缩，可知需要的比特位为 $mH(p_0)$，即实际传输的位数 $z=mH(p_0)$。$\frac{z}{m}$ 是要求的压缩率，记为 C，$C=\frac{z}{m}=H(p_0)$。

由 4.1 节的分析可知，当 $k=(m/n)\ln 2$ 时：

$$p_{\mathrm{err}}^{'}=(1-p_0)^k=(1-p_0)^{\frac{m}{n}(\ln p_0)}=(1-p_0)^{\frac{\ln p_0}{H(p_0)}(\frac{z}{n})}=$$

$$\exp(\frac{-(\ln p_0)\ln(1-p_0)}{-\mathrm{lb}p_0(p_0(\ln p_0)+(1-p_0)\ln(1-p_0))}\cdot\frac{z}{n})$$

当 $p_0=\frac{1}{2}$ 时：

$$p_{\mathrm{err\_min}}=\mathrm{e}^{\frac{1}{\mathrm{lbe}}\frac{z}{n}}=\mathrm{e}^{-\frac{z}{n}(\frac{\ln 2}{\ln 2\mathrm{be}})}=\mathrm{e}^{\frac{z}{n}\ln\frac{1}{2}}=0.5^{\frac{z}{n}}<0.618\ 5^{\frac{z}{n}}$$

即在原有基础上的最小假阳性率通过压缩后仍有减小的余地。表 1 是压缩型 Bloom Filter 应用的一个例子，说明通过压缩可以同时获得更少的信息交换量和更低的假阳性率。

表 1　压缩型 Bloom Filter 的应用实例

元素位数 m/n	元素传输位数 z/n	哈希函数个数 k	假阳性率 f
16	16.000	11	0.000 459
28	15.846	4	0.000 314
48	15.829	3	0.000 222

当不应用压缩时，即 m=z，则每个元素的位数为 16，此时最优的哈希函数个数为 11，假阳性率为 0.000 459；当应用压缩时，将每个元素的位数提高到 48，此时最优的哈希函数个数下降为 3 个，使假阳性率也下降到 0.000 222，每个元素的传输位数为 15.829，低于 16。

5　结束语

本文讨论了在 VANET 恶意节点检测过程中引入 Bloom Filter 机制，并对假阴性率和假阳性率进行了分析和论证，同时引压缩型 Bloom Filter，降低节点每次更新数据时的信息交换量，获得更高的检测速率和更低的假阳性率，进一步节省和降低通信开销、提高 VANET 的通信实时性。本文只针对车辆节点在通过基础设施对本地维护的恶意节点库进行更新的条件下，对基于伪名的 VANET 恶意节点检测过程进行讨论，并设计 Bloom Filter 在基于伪名的 VANET 恶意节点检测机制中的应用方案，但是仍然存在以下可能的情况需要进一步讨论和研究：(1)车辆节点在初始无路边基础设施提供 CA 的情况下，节点可能通过其他车辆节点进行恶意节点库更新；(2)在信号范围内无路边基础设施或在检测过程中由于节点的运动脱离了基础设施范围时的情况。对于第(1)种情况，由于节点本地的初始恶意节点库都由认证中心得到，因此可以利用 MAC 的特性，使任何车辆节点接收的恶意节点库不可篡改(即黑匣子)，从而保持节点库的权威性，免去重复认证和节点篡改数据库欺诈的风险；对于第(2)种情况，要注意传输中的恶意节点库的数据完整性，在此可以引用关系数据库中的“事务”的概念。今后将对上述 2 种情况的具体实现进行进一步研究和完善。

(下转第 136 页)