软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(9):3422−3436 [doi: 10.13328/j.cnki.jos.006294]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于统计量特征的数据库指纹方法*

方焱志, 黄煜坤, 彭煜玮

(武汉大学 计算机学院, 湖北 武汉 430072)

通信作者: 彭煜玮, E-mail: ywpeng@whu.edu.cn

摘　要: 在数据库中嵌入数字水印作为指纹是一种重要的数据库版权保护与身份溯源技术, 它对推动数据的共享融合起到了重要的保障作用. 针对现有数据库指纹方法在数据普适性上的不足开展研究, 提出了一种基于统计量特征的数据库指纹方法. 首先, 采用了迭代哈希的数据划分方法将数据划分为多个子集; 然后, 通过最优化算法将过滤极值后的数据子集特征最大/最小化, 根据基于最小错误率的贝叶斯决策计算得到最优阈值作为指纹信息. 通过理论分析验证了该方法的可行性与有效性, 同时也通过真实数据集上的实验结果证明了所提算法在抗攻击能力和普适性上的优势.

关键词: 数字水印; 数据库指纹; 统计量特征; 可用性约束; 鲁棒性水印

中图法分类号: TP311

中文引用格式: 方焱志, 黄煜坤, 彭煜玮. 基于统计量特征的数据库指纹方法. 软件学报, 2022, 33(9): 3422–3436. http://www.jos.org.cn/1000-9825/6294.htm

英文引用格式: Fang YZ, Huang YK, Peng YW. Database Fingerprinting Based on Statistical Features. Ruan Jian Xue Bao/Journal of Software, 2022, 33(9): 3422–3436 (in Chinese). http://www.jos.org.cn/1000-9825/6294.htm

Database Fingerprinting Based on Statistical Features

FANG Yan-Zhi, HUANG Yu-Kun, PENG Yu-Wei

(School of Computer Science, Wuhan University, Wuhan 430072, China)

Abstract: Digital watermarking to form fingerprints in databases is an important approach for database right protection and ownership identification. It provides protection for the sharing and fusion of data. As existing database fingerprinting methods have a deficiency in the universality of data, this study proposes a database fingerprinting approach based on statistical features. This approach first divides the host data into several subsets by an iterative hash function. Then, the statistical feature of each subset is maximized/minimized by an optimization algorithm after extreme values are filtered out. Finally, the optimum threshold is taken as fingerprint information which is calculated by Bayesian decision for minimum errors. This study also theoretically verifies the feasibility and effectiveness of the proposed method. The experimental results on real datasets demonstrate that the method has advantages in both robustness and universality.

Key words: digital watermarking; database fingerprinting; statistical features; availability constraints; robust watermarking

随着大数据时代的发展, 各类数据逐渐开始被发布在公共平台上用于共享或者被出售给消费者企业或个人进行深度应用. 在这种背景下, 被发布的数据也可以被视作一种产品——数字产品. 数字产品一旦被发布, 就面临着被非法拷贝给他人等不法行为的侵害. 这些非法拷贝的存在和传播对数字产品的所有权人的利益造成了巨大的威胁, 因此各种版权保护技术应运而生. 作为版权保护中非常重要的一种技术手段, 数字水印技术已经在各类数字产品的版权保护中被广泛使用[1], 如图像, 多媒体, 自然语言文本和地理矢量数据等. 而随着外包数据库 (outsourced database)[2]等概念的提出, 外包数据库的版权保护也开始成为数字水印技术研究的新领域.

自从 Agrawal 等人 2003 年首次提出基于修改最低有效位 (least significant bit, LSB)[3]的数据库水印算法后, 越来越多的数据库水印方法开始涌现出来, 包括: 基于位替换的水印算法[3,4], 可逆的水印算法[5], 基于数据统计量的

收稿时间: 2020-02-25; 修改时间: 2020-04-28, 2020-11-16; 采用时间: 2020-12-04; jos 在线出版时间: 2022-07-15

水印算法[6,7]等. 根据水印的健壮性, 这些数据库水印方法又可分为脆弱型和鲁棒型, 前者在受到微小的攻击后水印即被破坏, 适用于数据库的内容认证, 而后者可以容忍恶意攻击, 更多地被用于保护数据库的版权.

一种合格的鲁棒型数据库水印方案应该可以抵抗各种类型的恶意攻击, 攻击类型分为以下 5 种.

(1) 插入攻击: 攻击者向已嵌有水印的数据库中插入 n 个新元组;

(2) 删除攻击: 攻击者从已嵌有水印的数据库中删除 n 个元组;

(3) 修改攻击: 攻击者对已嵌有水印的数据库的 n 个元组进行修改;

(4) 组合攻击: 攻击者对已嵌有水印的数据库同时进行 (1)–(3) 组合而成的操作;

(5) 重嵌入攻击: 攻击者在已嵌有水印的数据库中嵌入新的水印.

而本文研究的数据库指纹技术则是水印技术的一个分支[8], 数据库指纹的目的是用标记识别数字产品分发过程中的接收者, 以便对流出的非法拷贝进行溯源[9]. 数据库指纹的原理是, 当同一个数据库被分发给不同的接收者时, 利用数据库水印算法在不同的数据库发布版本中分别嵌入不同的水印信息 (指纹). 当发现疑似盗版的数据库流出时, 可以提取其中的指纹信息并溯源, 从而确定泄露的来源.

在文献 [10] 中, Li 等人在密钥的基础上使用 MAC (message authentication code, 消息认证码) 和一个用户标识符来计算指纹信息. 同时, 作者提出使用元组属性的最高有效位来构造虚拟主键, 从而可以在没有主键的数据集中嵌入指纹. 但是, 虚拟主键对于每个元组可能并不唯一. 因此, 指纹的每一位可能无法被均匀地嵌入到数据中, 使得该方案对攻击的抵抗能力较弱. 在文献 [11] 中, Constantin 等人提出了针对查询优化的关系型数据库指纹技术, 其中使用了声明性语言来定义可用性约束, 通过搜索特定的模式 (如聚合和连接运算) 来优化水印嵌入, 但是该算法的指纹提取非常依赖于可用性约束, 数据集在受到攻击后指纹检测错误率会大大提高. Guo 等人 2006 年提出在两个级别上对数据库嵌入指纹[12], 在第 1 级被用来嵌入指纹的元组不会在第 2 级嵌入时被使用, 以防止冲突. 但是其在嵌入水印时, 直接对属性值二进制形式的第 j 位进行修改, 缺少对数据可用性的保障. 2007 年, Zhou 等人在文献 [13] 中提出 NoFiA, 一个用于关系型数据库防止非法拷贝与二次分发的水印框架. 基于该框架, 嵌入指纹的过程不会对原数据产生改动, 且可以轻松地管理指纹的相关参数. 2008 年, Wang 等人提出一种新的关系型数据水印算法, ATBaM[14]. 该方法将水印与人眼的视觉特性结合来保证水印的不可见性, 同时通过图像上的 Arnold 变换算法提高了鲁棒性. 2013 年, Jawad 等人利用基于差值扩展的遗传算法实现了可逆的数据库水印算法[15], 其通过遗传算法增大了水印容量并减小了引入的误差, 当水印容量的需求上升时, 遗传算法只需要在更大的特征空间中搜索最优解. 2014 年, Iftikhar 等人针对原有可逆水印算法在面对恶意攻击时不够健壮的缺陷, 在文献 [16] 中提出了 RRW, 一种数值型数据上鲁棒型的半盲可逆水印算法, 保证嵌入水印后的数据集在受攻击后, 仍能恢复得到大部分的原本数据集. 而 2019 年, Li 等人则提出一种低失真的可逆数据库水印方法[17]. 该方法通过计算数据集中每条数据值的预测误差, 得到预测误差的频率直方图, 对落在直方图频率最高处左右两侧数据进行局部偏移, 从而产生直方图的间隙, 并据此进行水印的嵌入提取. 但该方法对数值型数据的分布有特定要求, 要求预测误差的频率直方图在频率最高处附近尽可能分布集中, 且左右两侧数据频率差距较小. 针对现有方法存在的问题, 本文提出了一种基于统计量特征的数据库指纹方法, 其主要工作体现在以下方面.

(1) 通过分位点对数据集的分割, 消除了不同数据集的不同数据分布特征所带来的影响;

(2) 基于数据集的数据统计量嵌入指纹 (水印), 提高了算法的鲁棒性;

(3) 通过最优化算法, 在保证数据可用性约束的前提下, 尽可能提高数据集被攻击后的指纹提取能力;

(4) 同一个数据集可嵌入不同水印分发给不同对象, 可对数据集进行溯源;

(5) 引入基于最小错误率的贝叶斯决策, 降低了提取水印时的误解码率.

本文第 1 节从总体上介绍水印嵌入/提取的算法框架; 第 2 节和第 3 节分别详细地解释了水印嵌入和水印提取的算法; 第 4 节讨论了参数的选择与方法效果之间的关系; 第 5 节总结了与本文相关的工作, 并深入分析了其中两种最具代表性的水印方法; 第 6 节通过理论分析与实验与上述两种方法进行对比, 分析了算法的鲁棒性与效果; 第 7 节总结全文, 并对未来值得研究的方向进行探讨.

1 算法框架

本文提出的基于统计量特征的数据库指纹方法是一种半盲水印方法, 在水印检测时不需要原始数据集和水印信息, 而是需要一个嵌入阶段所生成的阈值. 图 1 给出了该水印方法的核心框架, 可划分为水印嵌入方法与水印提取方法两个部分.

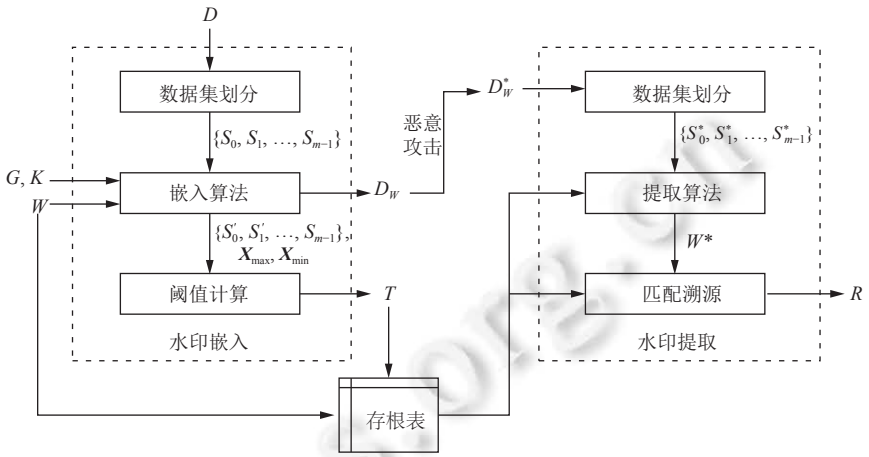


图 1 基于统计量特征的数据库指纹方法的框架

在基于统计量特征的数据库指纹方法中, 水印的嵌入以数据库中的关系表为单位进行. 因此, 水印嵌入方法的输入包括目标数据集 (或称数据表) D (P, A_0), 指定数据可用性约束 G 以及用户给定的密钥 K. 其中,

- 数据集 D 的模式中含有主键 P 和待嵌入数值型属性 A_0 (这里仅考虑单一属性, 多个目标属性的水印嵌入可以通过多次应用单属性水印嵌入来实现);
- 可用性约束 G 用于保证数据集可用性不受水印嵌入过程的影响, 它描述了属性 A_0 所有值允许的变化范围;
- 水印信息 W 为 0/1 构成的二进制序列, 通过随机起点的暴力搜索生成, 确保随机性和相异性.

水印嵌入算法利用密钥 K 将水印 W 嵌入 D 中, 得到嵌入水印的数据集 D_W 以及用于提取水印的阈值 T, 同时使得得到的 D_W 能够满足可用性约束 G. 水印嵌入完成后, 利用存根表保存阈值 T, 存根表的模式如表 1 所示. 嵌入水印后的数据集 D_W 随后就可以被发布给接收者.

表 1 存根表模式

字段	描述
watermark	二进制水印串
threshold	提取水印所用阈值
secret_key	划分时所用的密钥
partition_count	数据子集个数
info	数据接收者身份信息

水印嵌入的步骤如下:

(1) 水印生成: 根据用户指定的水印长度 l(二进制位数), 穷举长度为 l 的二进制水印串并计算其与已存在于存根表中的水印串之间的相似度, 若相似度均小于设定的临界相似度, 则采用生成的水印. 否则继续搜索, 直到查找到满足条件的水印串或者穷举完所有的二进制水印串. 临界相似度为超参数, 它确保已使用的水印之间不会过于相似, 以提高水印提取时的抗干扰能力;

(2) 数据集划分: 根据密钥 K, 将数据集 D 划分成 m 个互不相交的子集 $\{S_0, S_1, \ldots, S_{m-1}\}$;

(3) 水印嵌入: 在 G 的约束下, 基于数据集的统计量特征修改每个子集中的属性值, 来嵌入水印信息 W 中对应

于该子集的一位. 这一修改过程通过最优化算法来实现, 水印位的 0/1 对应最优化目标的最大值优化/最小值优化, 以保证水印鲁棒性和数据可用性之间的权衡. 水印串 W 与数据拥有者的身份信息被保存至存根表以用于未来的水印提取操作;

(4) 基于错误率的阈值计算: 以每个子集最优化得到的最大值/最小值和每个修改过后的子集作为输入, 计算提取水印时需要的阈值 T, 并将其保存至存根表.

嵌入水印后的数据集 D_W, 可能会被正常编辑或者恶意攻击 (以下统称为攻击), 例如: 子集删除攻击, 子集插入攻击, 子集修改攻击等. 受攻击后得到的数据集记为 D_W^*. 数据拥有者可利用水印提取算法根据对应于 D_W 的密钥 K 和提取阈值 T 从 D_W^* 中提取出水印 W^*, 进而根据 W^*来判定该数据集的版权归属.

水印提取的步骤如下:

(1) 数据集划分: 按照与水印嵌入中同样的划分算法将 D_W^* 划分成 m 个无交集的子集 $\{S_0^*, S_1^*, \ldots, S_{m-1}^*\}$;

(2) 基于阈值的水印提取: 对于子集, 基于数据集的统计量特征和阈值 T, 提取出每个子集对应的水印位;

(3) 基于投票的误差修正: 从每个子集提取出的水印位会经过多数投票, 修正水印信息被破坏而可能带来的误差, 从而得到最终提取出的水印串 W^*. 利用概率学的伯努利分布设置提取范围, 保证提取水印置信度在 95% 以上;

(4) 匹配溯源: 计算提取出的水印串 W^*和存根表中的水印串之间的海明距离[18]进行相似度匹配计算, 将最为匹配的数据接收者身份信息作为溯源结果 R.

2 水印嵌入

本节将介绍水印嵌入的细节以及嵌入过程中函数与参数选取的意义. 由于本文所提出的水印方法仅依赖于数据集 D 的主键 P 和目标属性 A_0, 且不修改除 A_0 以外的属性值, 在接下来的讨论中我们仅考虑主键 P 与属性 A_0. 同时, 为方便讨论, 将方法中涉及的符号标记总结于表 2 中.

表 2 符号标记

符号	描述
N	数据集元组总数
m	划分的子集数
W	水印位串
K	密钥
G_i	可用性约束
ref	分割参数
Θ	隐藏函数
α	分割下界比例
β	分割上界比例
T	提取水印用的阈值
$X_{\max}$	最大优化后的隐藏函数值集合
$X_{\min}$	最小优化后的隐藏函数值集合
S_i	数据子集
v_i	数据子集S_i对应的向量
l	嵌入的水印位长度
$\lvert S_i \rvert$	数据子集S_i的基数

2.1 数据划分

这一步骤的目的是通过哈希方法将数据的排列顺序打乱, 使攻击者在不知道哈希参数的情况下无法探知数据集中水印位与元组之间的隐藏关系, 因此无法针对性地进行攻击操作抹除水印, 从而达到增强抗攻击能力的目的.

数据划分阶段将根据密钥 K, 利用哈希取模的方式将数据集 $D(P, A_0)$ 划分成 m 个子集 $\{S_0, S_1, \ldots, S_{m-1}\}$. 对于

D 中的任意一个元组 r, 其所属子集的下标可按公式 (1) 计算.

$$partition(r) = H(K||H(r \cdot P||K)) \bmod m \tag{1}$$

其中, 密钥 K 为用户指定的私密字符串, 保证攻击者即使在掌握哈希算法的情况下也无法还原数据划分过程. 如算法 1 所示, 该划分方法的输入为元组的主键值与密钥 K 连接后的字符串, 通过加盐和迭代哈希[19]增加破解负担, 提高算法抗攻击能力. 数据划分阶段中可以使用任何哈希函数, 本文中使用的是 BKDR 哈希函数. 理想情况下, 划分形成的每个子集中包含的元组数为 $\frac{N}{m}$.

算法 1. dataset_partition.

输入: 数据集 D, 密钥 K, 分区数 m;
输出: 数据分区 $\{S_0,\ldots,S_{m-1}\}$.

for i **from** 0 **to** m–1
  $S_i = \{\}$
end for
for each tuple $r \in D$
  $partition(r)=H(K||H(r.P||K)) \bmod m$
  insert r into $S_{partition(r)}$
end for
return $\{S_0,\ldots,S_{m-1}\}$

关系型数据库中, 具有区分数据唯一性的主键值很少会被删除修改, 如公民的身份证号或学生学号等. 但如果主键值不具备数据分析价值 (如自增字段或随机生成的全局唯一 ID 等), 则主键值与水印目标属性 (A_0) 的相关性很低, 因此攻击者完全可以重建一套主键体系, 从而破坏数据集中嵌入的水印.

针对这种漏洞, 可以使用多个与数据集价值密切相关的属性组合代替主键 P, 如第 5 节实验中的数据集 D_1, 可以选取 (客户 ID, 交易时间, 供应商 ID) 作为主键 P. 这样在保证了唯一性的同时, 由于这些属性值承载了数据集的分析价值与商业价值, 攻击者无法替换或删除其中的属性, 保证了数据划分结果的稳定性.

2.2 求取分割界限

接下来的水印嵌入过程中需要一个分割界限, 它将每一个数据子集按照待嵌入水印数值属性的值切割成两部分, 将不规则数据集中的数据均匀分布在分割界限的两侧. 分割界限应满足以下性质:

(1) 良好的稳定性: 从嵌入水印前后的数据集中求得的分割界限应尽可能接近, 误差应当控制在一定范围内;

(2) 普适均匀的分割性: 分割界限应该把每一个数据子集分割成尽可能均匀的两部分.

求取分割界限的方法如算法 2 所示: 将数据集 D 按照待嵌入数值型属性值升序排序, 然后取出有序集合中序号在区间 $[0,\alpha \times N]$ 及 $[\beta \times N,N]$ 中的元组 ($0<\alpha<0.5, 0.5<\beta<1, \beta=1-\alpha$), 得到处理后的集合 D'. 取 D' 中待嵌入水印属性的均值作为分割参数 ref, α 与 β 具体取值将在第 6.5 节中详细讨论.

算法 2. get_ref.

输入: 数据集 D, 分割尺度 α, 集合大小 N;
输出: 分割参数 ref.

$D' = D.sort()$
$sum = 0$
$start = N * \alpha$
$end = N * (1-\alpha)$

for *i* **from** *start* **to** end−1

$sum+ = D'.getIndex(i)$

end for

ref=*sum*/(*end*−*start*)

return *ref*

由于分割参数 *ref* 是基于数据集的统计量计算得到, 当数据集较大且可用性约束较为严格时, 攻击操作对数据集统计量的修改可忽略不计[6], 因此 *ref* 能满足特性 (1). α 和 β 则保证了数据集首尾部分的极值能够被去除, 按此方法求得的 *ref* 接近于数据集中待嵌入属性的属性值中位数. 同时, 在哈希划分的随机性得到保障时, 可确保划分得到的每一个数据子集的数据分布接近于原始数据集, 因此所求得的 *ref* 对于每一个数据子集都具有均匀的分割性.

2.3 水印位嵌入

在本文所提出的水印方法中, 水印信息将被逐位地嵌入目标数据集中, 不同水印位的嵌入过程相互独立, 因此本小节仅讨论一个水印位的嵌入过程.

为保证抗攻击性, 在水印位嵌入时, 会将同一个水印位嵌入在多个数据子集中, 即使某些数据子集中嵌入的水印位因为攻击而损坏, 也能通过其他嵌入有同一水印位的数据子集进行恢复. 若水印长度为 l, 其第 t 个水印位 b_t 与嵌入 b_t 的数据子集 S_i 的对应关系为:

$$t = i \bmod l \tag{2}$$

对于数据子集 S_i, 水印位的嵌入过程仅涉及待嵌入属性 A_0. 因此, 每一个数据子集 S_i 可看作维度为$|S_i|$的向量 v_i, 向量第 j 维的值 v_{ij} 对应 S_i 中第 j 个元组的 A_0 属性值.

单个水印位嵌入算法的伪代码如算法 3 所示.

算法 3. encode_single_bit.

输入: 数据子集 S_i, 水印位 *bit*, 约束 G_i, 优化后的隐藏函数值集合 $X_{\max}$, $X_{\min}$;
输出: 修改后的数据子集 S_i^*.

if (*bit*==1)

$ps\_optimize(S_i, 1)$ s.t. to G_i

insert $\Theta\left(S_i^*\right)$ into $X_{\max}$

else

$ps\_optimize(S_i, 0)$ s.t. G_i

insert $\Theta\left(S_i^*\right)$ into $X_{\min}$

end if

return S_i^*

(1) 首先, 根据水印位 b_t, 对数据子集 S_i 对应的向量 v_i 做最优化处理. 本水印嵌入算法的思想是在一定的可用性约束条件下, 根据水印位 b_t 的取值 (0 或 1), 为向量 v_i 添加一个偏移向量, 使 v_i 通过隐藏函数变换后的值最小/最大, 取隐藏函数为:

$$\Theta(v_i + \Delta_i) = \frac{1}{n}\sum_{j=1}^{n} Sigmoid_{(\zeta,ref)}\left(v_{ij} + \Delta_{ij}\right) \tag{3}$$

$$Sigmoid_{(\zeta,ref)}(x) = \left(1 - \left(1 + \mathrm{e}^{\zeta(x-ref)}\right)\right)^{-1} \tag{4}$$

其中, 隐藏函数 $\Theta(v_i + \Delta_i)$ 的目的是将 n 维的向量 v_i 映射至 1 维. 映射方式是, 先将向量在每一维度上的取值映射为 1 或者 0. 若 $v_{ij} > ref$, 则映射为 1; 若 $v_{ij} < ref$, 则映射为 0. 然后将各个维度得到的映射值累加求和, 取平均数作为该向量的映射值. 为了实现这种二分映射, 本方法选取 $Sigmoid_{(\zeta,ref)}$ (公式 (4)) 作为二值映射函数, 该函数在

$\zeta>0$ 的情况下, 随 $x-ref$ 增大, 收敛于 1; 随 $x-ref$ 减小, 收敛于 0. 而5越大, 该函数在 0 点附近能更快地随 $x-ref$ 增大/减小收敛于 1/0.

(2) 对隐藏函数采用模式搜索算法, 搜索过程中受可用性约束 G_i 的限制[20]. 求出向量对应的最大最小隐藏函数值. 搜索结束后, $v_i^w=v_i+\Delta_i^*$ 为嵌入水印位后的向量, $x^i_{\max|\min}$ 为最优化后的隐藏函数值集合. 至此, 一位水印嵌入完毕.

模式搜索算法的伪代码如算法 4 所示. 可用性约束 G_i 通常可表现为函数形式, 例如 $G_i=\Delta_i-C$ (C 为常数) 可用来限制数据修改范围. 在模式搜索过程中, 将确保 $G_i>0$, 当 $G_i\leqslant 0$ 时, 停止搜索. 可用性约束的具体形式可由使用者根据嵌入数据的可用性要求与数据特点来实现.

隐藏函数的值表示了该数据子集中, 大于 *ref* 的值所占比例. 水印位的嵌入过程就是通过最大或最小优化将该比例增大或减小来实现水印位的嵌入.

算法 4. ps_optimize.

输入: 数据子集 S_i, 水印位 *bit*;
超参数: 终止步长 *min_length*, 搜索步长 *step_length*, 衰减率 *decay_rate*, 加速度 *accelearate*, 搜索轮数 *turn_num*;
输出: 修改后的数据子集 S_i^*.

```
end = S_i.size()
while ( step_length > min_len)
   accelerate_direction = [0,0,…,0]
   turn=0
   while (turn < turn_num)
      for j from 0 to end−1
         data = S_i.getIndex(j)
         new_data = data ± step_length
         if (bit==1)
            if (sigmoid (new_data) > sigmoid (data))
               data=new_data
               accelerate_direction[j] = ±1
            end if
         else
            if ( sigmoid(new_data) < sigmoid(data))
               data=new_data
               accelerate_direction[j] = ±1
            end if
         end if
      end for
      turn++
end while
   step_length = step_length + decay_rate
   S_i^* = S_i + accelerate * accelerate_direction
   update S_i^* if satisfies max or min according to the value of bit
end while
return S_i^*
```

2.4 解码阈值求解

解码阈值是对于嵌入后的数据集数据统计量特征的数值量化. 具体来说, 解码阈值量化了该数据集经过第2.3节所述的算法修改后所形成的隐藏函数值集合之间的分界点.

在模式识别的分类问题中, 错误率指将应属于某一类的模式错分到其他类的概率[21]. 本文所提出的水印方法的水印提取过程将待提取的数据子集对应的隐藏函数值与解码阈值比较, 以此判别其中所嵌入的水印位是0或1. 这可类比为二分类问题, 而解码阈值的求解决定了提取水印时的错误率, 解码错误概率 P_{err} 定义为:

$$\begin{aligned} P_{\text{err}} &= P(x<T \mid b_e=1)-P_1+P(x>T \mid b_e=0)-P_0 \\ &= P_1-\int_{-\infty}^{T} f(x \mid b_e=1)dx+P_0-\int_{T}^{+\infty} f(x \mid b_e=0)dx \end{aligned} \tag{5}$$

其中, T 为解码阈值, b_e 为提取出的水印位, P_1 为嵌入水印位为1的概率, P_0 为嵌入水印位为0的概率. 对该等式左右两边求导, 可得:

$$\frac{\partial P_{\text{err}}}{\partial T}=P_1\cdot f(b_e=1)-P_0\cdot f(b_e=0) \tag{6}$$

其中, $P_1=\dfrac{|X_{\min}|}{|X_{\min}|+|X_{\max}|}$, $P_0=1-P_1$. $f(b_e=1)$ 和 $f(b_e=0)$ 能通过卡方检验 (Chi-square)[22], 并且服从正态分布 $N(\mu_1,\sigma_1)$, $N(\mu_0,\sigma_0)$, 其中 $\mu_1,\sigma_1,\mu_0,\sigma_0$ 分别为最大最小优化隐藏函数阈值集合的均值与方差. 对于第6节中的测试数据集, 最大最小优化隐藏函数值的频率分布如图2所示.

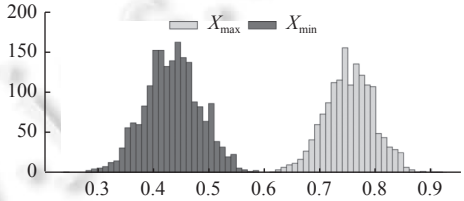


图2 最大最小优化隐藏函数值的频率分布

为了保证水印提取时的错误率最低, 解码阈值应取解码错误概率最小时的 T 值. 因此, 当一次导函数等于0且二次导函数大于零时, T 的值为最优的解码阈值. 其求导结果如下:

$$\frac{\partial P_{\text{err}}}{\partial T}=\frac{\sigma_0^2-\sigma_1^2}{2\sigma_0^2\sigma_1^2}T^2-\frac{\mu_1\sigma_0^2-\mu_0\sigma_1^2}{\sigma_0^2\sigma_1^2}T+\ln\left(\frac{P_0\sigma_1}{P_1\sigma_0}\right)+\frac{\mu_1^2\sigma_0^2-\mu_0^2\sigma_1^2}{2\sigma_6^2\sigma_1^2} \tag{7}$$

在实际操作过程中, 考虑到(7)式中的系数在方差较大的特殊情况下会由于浮点数精度而产生误差, 因此可以选择使用解码阈值的近似运算公式:

$$T=\frac{\mu_1+\mu_2}{2} \tag{8}$$

3 水印提取

水印提取的过程需要检测者提供嵌入水印时使用的密钥 K 以及待提取数据集 D_W^*, 解码阈值 T 等其他提取参数可从存根表中取得.

如算法5所示, 水印提取过程按照和第2.1节中相同的过程将 D_W^* 划分成 m 个子集 $\{S_0^*,S_1^*,\ldots,S_{m-1}^*\}$, 然后针对每一个子集提取出其中嵌入的水印位, 最后利用投票机制从这些水印位中确定提取出的水印串.

算法5. decode_watermark.

输入: 数据集 D_w, 子集数量 m, 密钥 K, 阈值 T*, 水印位长度 l;
输出: 检测到的水印 W_d.

ones[0,…,l−1] = [0,…,0]

$zeros[0,\ldots,l-1]=[0,\ldots,0]$

$S_0,\ldots,S_{m-1}=get\_partitions(D_W,K,m)$

for i **from** 0 **to** $m-1$

  $j=i \bmod l$

  $value=\Theta(S_i)$

  if $(value>T^*)$

    $ones[j]=ones[j]+1$

  else

    $zeros[j]=zeros[j]+1$

  end if

end for

for i **from** 0 **to** $l-1$

  if $(ones[i]>zeros[i])$

    $W_d[i]=$ ‘1’

  else if $(ones[i]<zeros[i])$

    $W_d[i]=$ ‘0’

  else

    $W_d[i]=$ ‘x’

  end if

end for

return W_d

3.1 水印位提取

采用第 2.2 节中相同的方法, 预处理数据集 D_w^* 求出待提取数据集的分割参数 *ref*. 然后根据 *ref* 求 S_i^* 的向量 v_i^* 对应的隐藏函数值, 将该值与存根表保留的解码阈值 T 进行比对, 若隐藏函数值大于 T, 则认为该子集中嵌入的水印位为 1, 否则认为嵌入的水印位为 0.

3.2 投票确认

为了提高水印算法的抗攻击性, 在数据划分时保证每一水印位能对应多个数据集划分, 即存在: $m>>l$. 因此, 最终水印信息中的每一个水印位应由其对应的多个子集的提取结果采用投票机制确认.

设水印位 b_i 对应数据集划分个数为 p, 其中提取出水印位为 0 的个数为 q, 提取出水印位为 1 的个数为 $p-q$. 传统的方法是: 当 $q>p-q$ 时, 投票机制确定该水印位为 0; 当 $q<p-q$ 时, 投票机制确定该水印位为 1; 当 $q=p-q$ 时, 认为该水印位提取失败. 但是, 即使数据集并未嵌入水印, 利用该方法也能从中提取出水印串[23], 从而对最终溯源判断造成干扰.

为解决这一问题, 本文所提的水印方法中引入了基于置信度的投票算法. 考虑数据集在未嵌入水印时, 其一个子集中能提取出水印位 0 或 1 的概率均为 1/2. 故从第 i 位对应的 p 个子集中, 假设提取出的 p 个水印位中有 k 个是有效的, 则将其视为一个成功 k 次, 失败 $p-k$ 次的伯努利实验. 其概率满足:

$$P(k)=C_p^k\left(\frac{1}{2}\right)^p=\frac{p!}{k!(p-k)!}\left(\frac{1}{2}\right)^p \tag{9}$$

现需要找到 k_0, 使得当 $k>k_0$ 时:

$$\sum_{i=k_0}^{p}C_p^i\left(\frac{1}{2}\right)^p\leqslant\varphi \tag{10}$$

其中, φ 为显著度参数. 当 φ 值足够小时, 即可认为是小概率事件, 这里按统计学经验可取 $\varphi = 0.05$. 即只要当 $k > k_0$ 时, 我们就有 95% 的把握认为该数据集嵌入了水印.

在基于置信度的投票确认水印后, 根据海明距离计算提取水印和存根表中各水印串之间的相似度, 若其中的最高相似度高于固定的相似度阈值, 该值的定义参考第 6.1 节, 则认为具有该最高相似度的水印串对应的数据接收者即为溯源结果.

4 参数选择

在基于统计量特征的数据库指纹方法中, 引入分割参数的意义在于, 在通过隐藏函数映射之前将不规则数据集中的数据均匀分布在 *ref* 的两侧, 便于之后利用模式搜索进行最大最小优化时能够增大不同类型优化的隐藏函数值的差距, 从而提高水印提取时的正确率.

分割参数 *ref* 有多种选取办法. 文献 [7] 中的方法对每一个数据子集 S_i 使用其均值 μ 与标准差 σ 计算得到 *ref*:

$$ref = \mu + c \times \sigma \tag{11}$$

其中, c 为秘密参数, 取值范围在 0 到 1 之间.

该方法在数据分布较为均匀的数据集中表现尚可, 但当数据集离散程度较大或存在极端值时, 该方法无法偏移数据集中心使其尽可能地分布在二分位点的两侧, 导致无法成功嵌入/提取水印. 在极端情况 (例如第 6 节中的数据集 D_1) 下, 由于方差较大, 该方法计算出的分割参数甚至超过数据集的值域范围, 因此该方法的普适性较差.

本文也尝试直接使用数据集的中位数作为 *ref* 对数据进行偏移. 该方法能够使偏移后的数据均匀地分布在二分界限两侧, 但该方法在后续的提取过程中, 需要使用原始数据集的每个子集的中位数, 不便于溯源信息的存储以及水印提取. 并且该方法对于数据偏移型攻击没有任何抵抗能力, 当嵌入水印后的数据被人为整体偏移一个较大值时, 与原数据子集的中位数比较便失去了意义.

而本文采取的 *ref* 参数选择方法兼顾了数据的离散性与存在极端值的情况, 并且具有良好的抗攻击能力. 本文的方法对整个数据集进行排序后剔除两端的极值, 取剩余部分的数据均值作为分割参数. 相比于前两种方法, 本文采用的方法得到的 *ref* 参数稳定性更强.

分割参数 *ref* 的效果取决于 α 、 β 值的大小. 对于一个数据集而言, 选取 α 、 β 的过大将导致处理后的数据集过小, 其数据统计量的稳定性降低, 对于修改攻击的抵抗能力下降; 选取的 α 、 β 过小则会导致无法完全剔除数据集中的极端值, 对于解码阈值的计算产生负面影响. 当原数据集足够大时, 按一定比例等量删除排序后的数据头部与尾部, 可以消除数据偏度带来的均值相对中位数值的差值影响, 因此, 此处令 $\beta = 1 - \alpha$. 我们将在第 6.5 节通过实验给出具体取值原则.

5 相关工作

现有基于数据统计量的水印算法可分类为: 基于位串的算法, 通常针对数值型数据, 通过改变浮点型数值的大小来嵌入 0/1 的水印位; 以及基于图像的算法, 通过人类视觉冗余的特性来增强算法的鲁棒性. 前者包括文献 [6,7,17,24,25] 等, 后者包括文献 [26–28] 等. 由于本文工作是基于位串实现, 故分别选择了基于位串的算法中, 最有代表性的文献 [7] 和文献 [17] 所提出的算法进行比较.

其中, 2008 年的文献 [7] 中提出了基于最优化算法的关系型数据水印技术, 其在文献 [6] 提出的基于数据统计量方法的基础上, 引入了遗传算法等最优化算法, 从而在数据修改范围的约束下最小化提取水印的误差. 在实验部分, 其主要与文献 [6] 使用的基于标记的划分算法进行了比对, 证明了其在抗攻击性上 (包括删除攻击、插入攻击和修改攻击) 的优越性. 但是由于其未对数据统计量指标的选取做出改进, 因而在数据集的标准差远大于均值的时候, 会出现抗攻击能力下降的现象

而 2019 年的文献 [17] 中提出了基于直方图间隙的水印方法 (HGW), 其对比了基于的差值扩展的水印算法 (DEW)[26], 结合遗传算法的差值扩展算法 (GADEW)[15], 预测误差值扩展水印算法 (PEEW)[26], 结合遗传算法的直方图偏移水印算法 (GAHSW)[29]. 该论文通过实验证明了, 相较于这 4 种算法, HGW 在抗删除攻击、修改攻击上

体现出良好的稳定性. 其引入了预测误差, 即数据相较于数据范围中间值的差的绝对值. 从而根据数据集中每条数据预测误差的频率分布来制定水印嵌入提取方案, 该频率分布对于数据的删除、修改操作相对不敏感, 最终提取水印的正确率能稳定在某个范围. 整体来看, 该方法在水印嵌入提取的思路上, 与我们的方法有类似性, 都是采集数据集的某个统计量特征. 但是, 基于直方图间隙的数据集对于数据的分布有一定要求. 当数据分布于数据集的极值或数据集中值附近, 会导致生成的预测误差直方图中, 最高频率点出现在直方图的最右侧或最左侧, 因而无法通过偏移最高频率左右两侧的预测误差频率来表示 0/1 水印位; 此外, 当嵌入水印属性的数值分布较为离散, 会导致生成的预测误差直方图中直方图间隙较多, 若间隙正好出现在最高频率点左或右侧, 则同样无法通过偏移表示水印位. 故相比本文提出的算法而言, HGW 算法在数据集兼容性上有所欠缺.

因此, 在第 6 节实验部分, 我们将选择文献 [7] 和文献 [17] 提出的算法进行对比.

6 实验结果

这本节通过实际数据集对本文所提出水印方法的效果进行测试. 水印方法使用 Java 语言实现[30], 实验数据集存放在 MySQL 5.7 中. 实验运行在一台搭载有 Intel Core i5-6300HQ 2.30 GHz 以及 8 GB 内存的 PC 机上.

实验数据集 D_1 来自中农网[31]2013 年的农贸交易数据, 共 211 549 个元组, 以 OFLATLOSE (转让盈亏, 浮点型数据) 作为水印嵌入属性, 其均值为–93.516, 标准差为 4 168.543. 实验数据集 D_2 来自 MetaTrader4 数据中心的英镑-美元外汇交易数据[32], 共 61 861 个元组, 以成交量 (整型数据) 作为水印嵌入属性, 其均值为 29.857, 标准差为 16.180. 如图 3 和图 4 所示, 这两个数据集有着不同的数据分布特征.

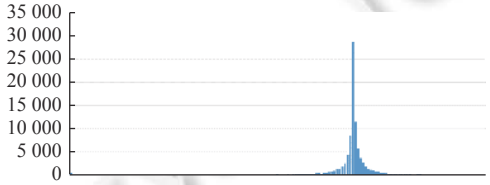


图 3 数据集 D_1 的频数分布直方图

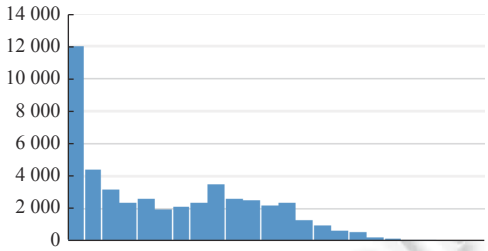


图 4 数据集 D_2 的频数分布直方图

来自不同领域的数据集具有不同的数据分布特征, 例如农业大数据具有周期性[33]等特点. 而只有适用于各领域, 各数据分布特征的水印算法才更具有实际应用价值.

在后续的抗攻击实验中, 我们将使用的比对指标为溯源匹配率, 对比文献 [17] 提出的使用比特错误率 (bit error rate, BER)——即错误的位数与水印总位数的比率来衡量算法的鲁棒性, 此处结合了数据库指纹分发给多个目标的特性. 同时在文献 [17] 中原始算法的实现中, 没有考虑提取结果除 0/1 外, 还应当设置提取失败的标志位, 即未嵌入水印的数据集仍然可能提取出某个特定的水印串, 故实验中我们额外对文献 [17] 添加了基于置信度的投票机制.

6.1 指纹容量

这一实验测试在未经恶意攻击时一个数据集最多能分发给多少个不同的对象, 即指纹的最大容量. 生成二进制水印串时, 必须保证生成的水印串与存根表中同一数据集对应的水印串的相似度低于临界相似度. 因此, 指纹容量与水印串长度以及临界相似度的设置相关. 而临界相似度的确定由分发对象的数量决定. 当分发对象较少时, 临界相似度可尽量的低, 使得因为两个相似度较高的水印串导致错误匹配事件发生的概率降低. 例如只有两个分发对象时, 临界相似度可定义为 0, 即两个指纹水印的每一位都两两不同. 但临界相似度降低的同时, 指纹容量也会减少, 导致不能分发给足够多的对象. 因此, 此处可根据实际需要权衡错误匹配概率与指纹容量, 选择固定的临界相似度. 接下来的攻击实验均在已分发指纹数为 3, 水印串长度为 16 且临界相似度为 0.7 的条件下进行.

指纹容量测试采用第 1.1 节中所述的方法不断重复水印生成过程, 并将生成结果保存至存根表中, 直至水印空间被搜索完毕. 由于在生成过程中, 确保新生成的水印与已存在的水印之间相似度不大于设定的临界相似度, 因此可以保

证当水印空间搜索完毕时, 最终存入存根表的所有水印之间的相似度都控制在临界相似度之下. 此时, 存根表的水印数量即为水印空间的水印容量. 本实验记录在临界相似度固定的条件下, 水印空间的水印容量与水印长度之间的关系.

由图 5 的实验结果可见, 指纹容量随水印串长度增加, 整体呈现上升趋势, 但存在周期性的极小值. 实际使用中, 建议选取周期极大值对应的水印串长度作为嵌入水印串长度, 提高单位水印串利用率.

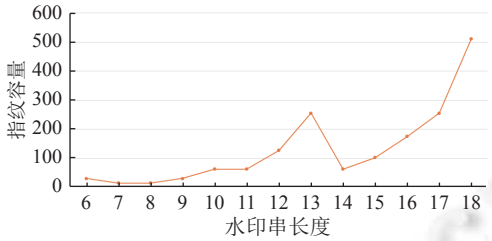


图 5 水印串长度对指纹容量的影响

6.2 插入攻击

在插入攻击实验中, 我们通过在数据集插入不同比例的重复冗余元组, 然后再提取水印评估对插入攻击的抵抗能力. 由于本文提出的水印方法使用基于组合主键的哈希数据划分算法, 且嵌入水印的对象是划分所得子集的统计特征, 故插入元组的攻击手段对其影响几乎没有. 实验结果也表明, 本水印算法, 以及文献 [7] 与文献 [17] 均对插入攻击表现良好. 在插入元组比例 0~100% 的条件下, 对于随机数据插入、均匀分布数据插入、单一数据插入的攻击, 3 种方法最终提取水印的匹配率均为 100%, 故此处不再给出对比图表.

6.3 删除攻击

假定攻击者按一定比例 γ 随机删除已嵌入水印数据集中的元组, 设数据集中原有 n 个元组, 这一实验从中随机选取规模为 $n\cdot(1-\gamma)$ 的子集, 然后从子集中提取水印并计算与原始水印信息的溯源匹配度. 溯源匹配度的计算方法是, 首先计算提取出的水印串 W^*和存根表中的水印串之间的海明距离, 再用海明距离除以水印串长度得到相异度, 最后用 1 减去相异度即可得到溯源匹配度.

由于攻击者不知道密钥 K, 而数据划分步骤能充分保证划分得到的子集在分布上的分散性和均匀性, 因此每个子集可视为被删除了 $\frac{n\cdot\gamma}{p}$ 个元组. 显然, 数据集的元组数越多, 划分得到的子集越多, 水印位嵌入时的冗余就越大, 因而水印的抗删除攻击能力越强.

图 6 与图 7 分别呈现了在数据集 D_1 和 D_2 上的删除攻击测试.

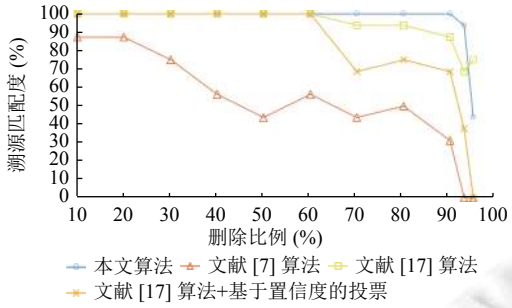


图 6 数据集 D_1 上的删除攻击测试

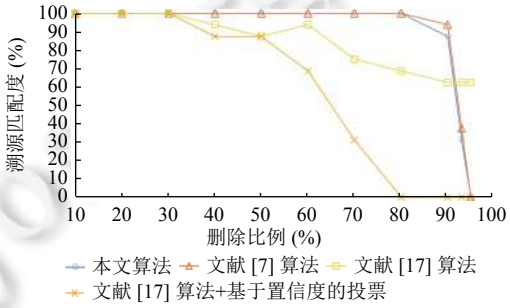


图 7 数据集 D_2 上的删除攻击测试

由图 6 可观察到, 在数据集 D_1 上, 文献 [7] 提出的算法表现最差, 印证了在数据集的标准差远大于均值的时, 该算法抗删除攻击性能较弱. 而文献 [17] 算法的原始实现与本文算法表现相差无几, 甚至在删除比例为 95% 时效果更好, 这是因为在没有引入基于置信度的投票机制的情况下, 即使失去了原本嵌入的水印信息, 也能提取出特定的随机水印串, 那么该匹配度的大小也失去了意义. 因此, 在引入基于置信度的投票机制后, 文献 [17] 算法在删除 95% 后溯源匹配度就达到了 0.

由图 7 可观察到, 在数据集 D_2 上, 本文算法与文献 [7] 提出的算法表现均为最佳, 文献 [17] 的算法表现很差. 这是由于文献 [17] 提出的算法, 与预测误差的频率直方图中频率最高处左右两侧频率之和的比例有关, 但数据集 D_2 的数据在左端分布较为集中, 导致左右两侧频率之和的差距过大, 使得水印嵌入过程中难以修改频率分布特征以表现出水印位 0/1 对应的频率分布特征. 而此处本文算法之所以在删除比例达到 95% 时, 溯源匹配度会降到 0, 是因为剩余元组数过少 (数据集 D_2 在删除 95% 后仅剩 3 093 条元组), 在哈希划分后, 每个数据子集的统计特征与原数据整体统计特征无法保持一致, 即提取结果近似于随机的水印串, 故在第 3.2 节的投票确认中被判定为未嵌入水印.

6.4 修改攻击

由于非法盗用数据集的攻击者是希望利用蕴含价值的数据获利, 那么其对数据集进行的改动也必然在数据可用性的约束范围之内. 为简化修改攻击流程, 这一实验的攻击手段为: 以一定比例 δ ($0.2\% \leqslant \delta \leqslant 2\%$) 随机修改已嵌入水印的属性值. 此处实验与文献 [17] 中的修改攻击实验设计不同, 文献 [17] 中设计的自变量为修改的元组数量比例, 而本文是固定修改元组比例为 100% (即数据集所有元组), 自变量为属性值随机变化的比例上限. 由于随机修改存在一定的偶然性, 故本实验记录的水印匹配度为同一个嵌入指纹的数据集分别进行 5 次不同的随机修改后再提取水印测得的平均值.

图 8、图 9 展现了算法在数据集 D_1, D_2 抵抗修改攻击的效果. 可以发现基于统计量的水印方法对于修改攻击均表现良好. 因为一定范围内的随机修改不会对数据分布造成较大波动, 但当修改比例过大时, 会使本方法在嵌入阶段求得的解码阈值无法用于修改后数据集的水印提取. 实验结果看出, 本文算法相较于文献 [7]、文献 [17] 的算法在数据集 D_1 和 D_2 上的抗修改攻击表现更好. 初步分析, 这是由于文献 [7] 算法使用的数据集统计特征中, 包含整体数据集的方差, 随机修改则会使得该项产生较大变化, 导致溯源匹配度随修改攻击比例增大而下降; 文献 [17] 所提出的算法中, 水印嵌入与预测误差的频率直方图有关, 具体来说, 直方图中的两条直方数据间隙会影响水印的嵌入提取, 修改数据会导致原本的直方图分布中的直方间隙产生变化, 因而抗修改攻击效果不理想; 而我们的算法的分割界限是由数据集的去极值后的均值决定, 这种小范围内的随机修改攻击对该均值影响不大, 因此相比前两种方法在抗修改攻击中取得较好效果.

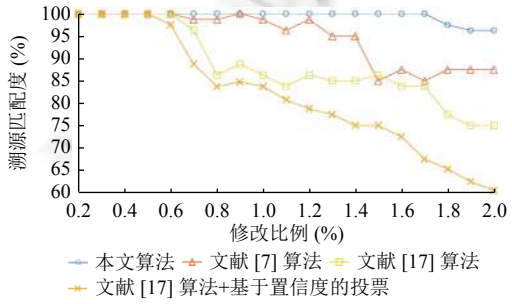


图 8 数据集 D_1 上的修改攻击测试

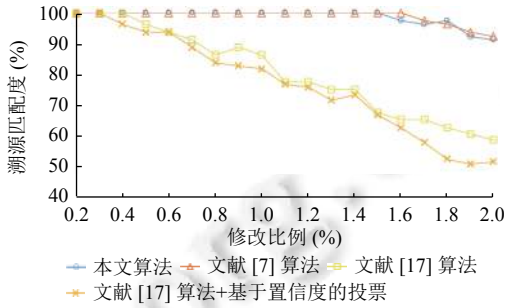


图 9 数据集 D_2 上的修改攻击测试

6.5 分割界限

为研究数据分割界限对水印提取效果的影响, 我们以数据集 D_1 为水印载体, 通过改变 α 取值, 来观察数据分割界限对最终溯源匹配度的影响, 实验结果如图 10 所示.

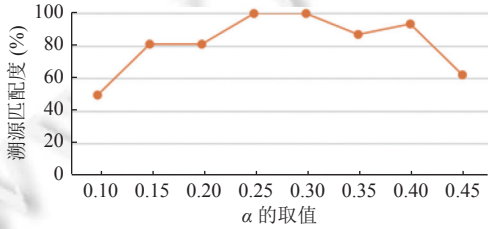


图 10 α 取值对指纹提取效果的影响

可以看出, 本文提出的水印方法在$\alpha \in (0.2,0.35)$时, 对于数据集 D_1 的溯源匹配度较高. 实际使用中, 建议采用四分位点附近的值作为α、β的取值.

7 总结与展望

本文提出了一种基于统计量特征的数据库指纹方法. 该方法借助统计量, 将数据子集的特征根据分割参数 *ref* 映射为水印位的 0 或 1, 同时使用最优化算法, 在不影响数据可用性的前提下, 将数据子集的特征最大最小化, 并根据贝叶斯最小错误率, 计算得到最优阈值, 降低水印位被错误判别的概率. 此外, 本文提出了一种数据分割机制, 该机制剔除了数据集中的极端值, 从而得到一个普适的分割参数 *ref*, 对同一个数据集中的每一个数据子集都达到较为均匀的二分效果. 通过该机制, 能够增加本方法的普适性, 使得本方法对于不同数据分布类型的数据集都有较强的抗攻击能力. 最后, 本文通过真实数据集上的实验验证了本文方法的有效性.

未来的研究包括: (1) 根据数据的使用目的来保证数据的失真最小; (2) 适用于更多数据类型的水印技术; (3) 进一步提高算法的效率以及抗攻击能力.

References:

[1] Abdel-Hamid AT, Tahar S, Aboulhamid E. A survey on ip watermarking techniques. Design Automation for Embedded Systems, 2004, 9(3): 211–227. [doi: 10.1007/s10617-005-1395-x]

[2] Hacigumus H, Iyer B, Mehrotra S. Providing database as a service. In: Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE, 2002. 29–38. [doi: 10.1109/ICDE.2002.994695]

[3] Agrawal R, Haas PJ, Kiernan J. Watermarking relational data: Framework, algorithms and analysis. The VLDB Journal, 2003, 12(2): 157–169. [doi: 10.1007/s00778-003-0097-x]

[4] Agrawal R, Haas PJ, Kiernan J. A system for watermarking relational databases. In: Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. San Diego: ACM, 2003. 674. [doi: 10.1145/872757.872865]

[5] Zhang Y, Yang B, Niu XM. Reversible watermarking for relational database authentication. Journal of Computers, 2006, 17(2): 59–66.

[6] Sion R, Atallah M, Prabhakar S. Rights protection for relational data. IEEE Trans. on Knowledge and Data Engineering, 2004, 16(12): 1509–1525. [doi: 10.1109/TKDE.2004.94]

[7] Shehab M, Bertino E, Ghafoor A. Watermarking relational databases using optimization-based techniques. IEEE Trans. on Knowledge and Data Engineering, 2008, 20(1): 116–129. [doi: 10.1109/TKDE.2007.190668]

[8] Milano D. Content control: Digital watermarking and fingerprinting. White Paper, Rhozet, a Business Unit of Harmonic Inc., 2012.

[9] Wang CD, Yang L, Wan FJ, Liu ZL, Zhu LK. Survey on database watermarking models and algorithms. Acta Electronica Sinica, 2019, 47(4): 946–954 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2019.04.022]

[10] Li YJ, Swarup V, Jajodia S. Constructing a virtual primary key for fingerprinting relational data. In: Proc. of the 3rd ACM Workshop on Digital Rights Management. Washington: ACM, 2003. 133–141. [doi: 10.1145/947380.947398]

[11] Constantin C, Gross-Amblard D, Guerrouani M. Watermill: An optimized fingerprinting system for highly constrained data. In: Proc. of the 7th Workshop on Multimedia and Security. New York: ACM, 2005. 143–155. [doi: 10.1145/1073170.1073196]

[12] Guo F, Wang JM, Li DY. Fingerprinting relational databases. In: Proc. of 2006 ACM Symp. on Applied Computing. Dijon: ACM, 2006. 487–492. [doi: 10.1145/1141277.1141391]

[13] Zhou M, Wang JM, Wang CK, Li DY. A novel fingerprinting architecture for relational data. In: Proc. of 2007 Inaugural IEEE–IES Digital EcoSystems and Technologies Conf. Cairns: IEEE, 2007. 477–480. [doi: 10.1109/DEST.2007.372022]

[14] Wang CK, Wang JM, Zhou M, Chen GS, Li DY. ATBaM: An arnold transform based method on watermarking relational data. In: Proc. of the 2008 Int'l Conf. on Multimedia and Ubiquitous Engineering. Busan: IEEE, 2008. 263–270. [doi: 10.1109/MUE.2008.45]

[15] Jawad K, Khan A. Genetic algorithm and difference expansion based reversible watermarking for relational databases. Journal of Systems and Software, 2013, 86(11): 2742–2753. [doi: 10.1016/j.jss.2013.06.023]

[16] Iftikhar S, Kamran M, Anwar Z. RRW—A robust and reversible watermarking technique for relational data. IEEE Trans. on Knowledge and Data Engineering, 2015, 27(4): 1132–1145. [doi: 10.1109/TKDE.2014.2349911]

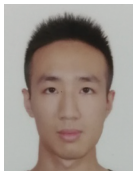
[17] Li Y, Wang JW, Ge SK, Luo XY, Wang B. A reversible database watermarking method with low distortion. Mathematical Biosciences and Engineering, 2019, 16(5): 4053–4068. [doi: 10.3934/mbe.2019200]

[18] Bookstein A, Kulyukin VA, Raita T. Generalized hamming distance. Information Retrieval, 2002, 5(4): 353–375. [doi: 10.1023/A:

1020499411651]
[19] Biham E, Dunkelman O. A framework for iterative hash functions—HAIFA. Santa Barbara: 2007.
[20] Lewis RM, Torczon V. Pattern search methods for linearly constrained minimization. SIAM Journal on Optimization, 2000, 10(3): 917–941. [doi: 10.1137/S1052623497331373]
[21] Sun JX. Modern Pattern Recognition. 2nd ed., Beijing: Higher Education Press, 2008. 132–137 (in Chinese).
[22] McHugh ML. The chi-square test of independence. Biochemia Medica, 2013, 23(2): 143–149. [doi: 10.11613/BM.2013.018]
[23] Niu XM, Zhao L, Huang WJ, Zhang H. Watermarking relational databases for ownership protection. Acta Electronica Sinica, 2003, 31(S1): 2050–2053 (in Chinese with English abstract). [doi: 10.3321/j.issn:0372-2112.2003.z1.024]
[24] Deshpande A, Gadge J. New watermarking technique for relational databases. In: Proc. of the 2nd Int'l Conf. on Emerging Trends in Engineering & Technology. Nagpur: IEEE, 2009. 664–669. [doi: 10.1109/ICETET.2009.160]
[25] Huang KY, Yue M, Chen PF, He YS, Chen XY. A cluster-based watermarking technique for relational database. In: Proc. of the 1st Int'l Workshop on Database Technology and Applications. Wuhan: IEEE, 2009. 107–110. [doi: 10.1109/DBTA.2009.38]
[26] Farfoura ME, Horng SJ. A novel blind reversible method for watermarking relational databases. In: Proc. of Int'l Symp. on Parallel and Distributed Processing with Applications. Taipei: IEEE, 2010. 563–569. [doi: 10.1109/ISPA.2010.63]
[27] Odeh A, Al-Haj A. Watermarking relational database systems. In: Proc. of the 1st Int'l Conf. on the Applications of Digital Information and Web Technologies. Ostrava: IEEE, 2008. 270–274. [doi: 10.1109/ICADIWT.2008.4664357]
[28] Zhang ZH, Jin XM, Wang JM, Li DY. Watermarking relational database using image. In: Proc. of the 2004 Int'l Conf. on Machine Learning and Cybernetics. Shanghai: IEEE, 2004. 1739–1744. [doi: 10.1109/ICMLC.2004.1382056]
[29] Hu DH, Zhao D, Zheng SL. A new robust approach for reversible database watermarking with distortion control. IEEE Trans. on Knowledge and Data Engineering, 2019, 31(6): 1024–1037. [doi: 10.1109/TKDE.2018.2851517]
[30] https://github.com/F-ca7/AsterMark
[31] https://www.zhongnongwang.com
[32] http://dataju.cn/Dataju/web/datasetInstanceDetail/43
[33] Wu CY, Wu CW, Xiong YL, Tao PY. Overview of agriculture big data research. Modern Agricultural Science and Technology, 2017, (17): 290–292, 295 (in Chinese with English abstract). [doi: 10.3969/j.issn.1007-5739.2017.17.167]

附中文参考文献:

[9] 王春东, 杨蕾, 万福金, 刘哲理, 朱立坤. 数据库水印模型及其算法综述. 电子学报, 2019, 47(4): 946–954. [doi: 10.3969/j.issn.0372-2112.2019.04.022]
[21] 孙即祥. 现代模式识别. 第2版, 北京: 高等教育出版社, 2008. 132–137.
[23] 牛夏牧, 赵亮, 黄文军, 张慧. 利用数字水印技术实现数据库的版权保护. 电子学报, 2003, 31(S1): 2050–2053. [doi: 10.3321/j.issn:0372-2112.2003.z1.024]
[33] 吴重言, 吴成伟, 熊燕玲, 陶佩莹. 农业大数据综述. 现代农业科技, 2017, (17): 290–292, 295. [doi: 10.3969/j.issn.1007-5739.2017.17.167]



方焱志(1998－), 男, 学士, 主要研究领域为数据库, 数据库水印.

彭煜玮(1980－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为数据库, 数据库水印.

黄煜坤(1999－), 男, 学士, 主要研究领域为数据库, 数据库水印.