软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(11):3571−3587 [doi: 10.13328/j.cnki.jos.005812]

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

分级可逆的关系数据水印方案*

侯瑞涛[1,2], 咸鹤群[1,2], 李 京[1], 狄冠东[1]

[1](青岛大学 计算机科学技术学院,山东 青岛 266071)

[2](信息安全国家重点实验室(中国科学院 信息工程研究所),北京 100093)

通讯作者: 咸鹤群, E-mail: xianhq@126.com

摘 要: 关系数据可逆水印技术是保护数据版权的方法之一.它克服了传统的关系数据数字水印技术的缺点,不仅可以声明版权,而且可以恢复原始数据.现有方法在恢复原始数据时不能控制数据恢复的程度,无法调节数据的可用性.提出了一种分级可逆的关系数据水印方案,定义了数据质量等级来反映水印嵌入对数据可用性的影响,设计了用于实现分级可逆水印的分区嵌入、等级检测、水印检测以及等级提升算法.数据所有者在数据分发前预先设定若干数据质量等级,以数据分区为单位嵌入水印.每个数据分区使用独立的密钥控制水印信息的位置和取值.如果数据使用者希望提升当前数据的可用性,可向数据所有者申请或购买相关密钥,提升当前数据的数据质量等级.对于任意数据质量等级的数据,其中的数字水印均可用于证明版权.采用分区的辅助数据,实现了灵活的水印逆操作.设计了有效的哈希表冲突解决方法,降低了计算和存储开销,提高了该方案的实用性.实验结果显示,方案具有良好的计算性能以及鲁棒性,可满足现实应用场景的需求.

关键词: 水印;分级可逆;版权;关系数据

中图法分类号: TP309

中文引用格式: 侯瑞涛,咸鹤群,李京,狄冠东.分级可逆的关系数据水印方案.软件学报,2020,31(11):3571−3587. http://www.jos.org.cn/1000-9825/5812.htm

英文引用格式: Hou RT, Xian HQ, Li J, Di GD. Graded reversible watermarking scheme for relational data. Ruan Jian Xue Bao/ Journal of Software, 2020,31(11):3571−3587 (in Chinese). http://www.jos.org.cn/1000-9825/5812.htm

Graded Reversible Watermarking Scheme for Relational Data

HOU Rui-Tao[1,2], XIAN He-Qun[1,2], LI Jing[1], DI Guan-Dong[1]

[1](College of Computer Science and Technology, Qingdao University, Qingdao 266071, China)

[2](State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences), Beijing 100093, China)

Abstract: Reversible watermarking technique for relational data is intended to protect the copyright. It overcomes the shortcomings of traditional watermarking techniques. It can not only claim the copyright of data, but also recover the original data from the watermarked copy. However, existing reversible watermarking schemes for relational data cannot control the extent of data recovery. Aiming at this problem, a graded reversible watermarking scheme for relational data is proposed in the study. Data quality grade is defined to depict the impact of watermark embedding on the usability of data. Watermark embedding, grade detection, watermark detection, and grade enhancement algorithms are designed to achieve graded reversibility of watermark. Before distributing the data, the data owner can predefine several data quality grades, then embed the watermark into data partitions. A unique key is used in each data partition to control the position and value of the watermark information. If data users are not satisfied with the usability of data, they can require or purchase

* 基金项目: 山东省自然科学基金(ZR2019MF058); 国家自然科学基金(61702294); 信息安全国家重点实验室开放课题(2020-MS-09)

Foundation item: Natural Science Foundation of Shandong Province (ZR2019MF058); National Natural Science Foundation of China (61702294); Open Program of the State Key Laboratory of Information Security (2020-MS-09)

收稿时间: 2017-11-09; 修改时间: 2018-02-03, 2018-05-06, 2018-11-17; 采用时间: 2018-12-28

relevant keys from the owner to upgrade the data quality grade. The watermark in relational data with any data quality grade is sufficient to prove the copyright. Flexible watermark reversion is achieved via partitioned auxiliary data design. A more practical mechanism is devised to efficiently handle the hash table collision, which reduces both computational and storage overhead. Experiments on algorithms and watermark show that the proposed scheme is feasible and robust.

Key words: watermark; graded reversible; copyright; relational data

随着大数据时代的到来,数据已经渗透到各行各业,成为重要的生产因素[1].数据价值的不断提升,让人们越来越关心版权保护问题.关系数据是常用的数据类型之一,普遍用于数据分析和数据交流.某些人可能在未经授权的情况下对数据进行恶意复制来获取经济利益,因此,关系数据的版权保护问题成为该领域的研究热点之一.

数字水印技术作为信息隐藏技术的一个重要分支,是实现版权保护的有效方法[2].数字水印技术最初被应用于音频、图像和视频等多媒体数据的版权保护[3−6].2002 年,IBM Almaden 研究中心的 Agrawal 等人以及美国 Purdue 大学的 Sion 等人结合关系数据的特点,提出了关系数据数字水印技术[7,8].该技术的基本原理是将一些标识信息添加到原数据中,在不影响数据正常使用的前提下,达到声明版权的目的.这些标识信息构成了数字水印,但是标识信息的嵌入势必会导致永久性的数据失真,对数据可用性造成永久性损害.如果数据可用性无法满足使用者的要求,水印嵌入将导致数据失去使用价值[9].针对上述问题,关系数据可逆数字水印技术开始受到研究者们的广泛关注.

关系数据可逆水印技术不仅可以执行水印嵌入操作,还可以执行水印嵌入的逆操作,即将嵌入到数据中的水印去除,实现数据恢复.但是现有的关系数据可逆水印技术在实际应用中仍存在问题.假设 Alice 为数据所有者,Bob 为使用者.Alice 为证明对数据的版权,在出售或分发数据前,使用关系数据可逆水印技术对数据进行水印嵌入.Bob 在使用该数据时发现其可用性无法满足他的要求,便向 Alice 申请或购买相关密钥,执行水印嵌入的逆操作,消除数据中的水印,提高数据的可用性.此时,若 Bob 在未经 Alice 授权的情况下将恢复后的数据出售给他人,Alice 将无法从该数据中检测出水印,证明数据的版权.根据上述的应用场景,现有的关系数据可逆水印技术虽可以恢复原始数据,将数据可用性提升到水印嵌入之前的程度,但所有者将不能继续证明对该数据的版权.出现该问题的原因是:现有的关系数据可逆水印技术对数据恢复的程度缺乏控制,只能将数据中的水印全部去除.因此,如何在提升数据可用性的同时实现数据的版权保护成为新的研究方向.

本文针对现有的关系数据可逆水印方案中存在的问题,提出了一种分级可逆的关系数据水印方案,定义了数据质量等级来反映数据的可用性,设计了用于实现分级可逆水印的分区嵌入、等级检测、水印检测以及等级提升等算法.数据所有者对数据进行分发或出售前,先使用水印分区嵌入算法对数据进行水印嵌入,该过程在嵌入水印的同时,会预设若干数据质量等级.如果数据使用者对当前数据的可用性不满意,可先检测当前数据质量等级,然后向数据所有者申请或购买相关密钥,从而去除部分数据分区中的水印,对数据质量等级进行提升,按需增强数据的可用性.此后,如果数据的使用者在未经过所有者授权的情况下分发该数据,所有者仍然可以从数据中检测出剩余的水印,达到版权声明的目的.方案的主要创新点如下.

(1) 提出了分级可逆的关系数据水印方案.通过将数据划分为若干个数据分区,实现以分区为单位的水印嵌入,各数据分区中的水印均可声明版权.

(2) 定义数据质量等级来反映数据的可用性,通过对数据分区中的水印执行逆操作,实现数据质量等级的提升,灵活调节数据可用性.

1 相关工作

关系数据数字水印技术由 Agrawal 等人首次提出,其基本原理是在某些特定的元组、特定的属性值以及特定的比特位包含特殊值,这些特殊值的组合构成数字水印[7].特定位置和特殊值的选择,由密钥决定.水印检测时只需提供密钥,不需要原始数据.Sion 等人将关系数据分割为大小相同的分区,而水印的嵌入则是通过改变分区的分布特性实现的[8,9].但是该方案采用的数据分区方法难以抵御元组删除攻击和元组添加攻击[9].牛夏牧等人提出一种可将具有实际意义的字符串嵌入到关系数据中的水印方案[10],其本质是在关系数据的属性编号和最

低有效位之间嵌入一种特殊的匹配规则[9,10].但方案只能验证数据中是否嵌有这种匹配规则,而无法检测出真正的水印信息.张志浩等人则将一幅图像编码成水印,将其嵌入到关系数据中[11].其原理是:将数据分成与图像同等大小的数据块,在数据块的每个属性值中嵌入图像的一个像素点.向玥等人则针对主键攻击问题提出了基于虚拟主键的关系数据水印嵌入方法[12],该方法通过(7,4)汉明码和大数表决的结合,增强了水印的抗攻击能力.周刚等人提出一种基于改进型 C4.5 算法的关系数据零水印模型[13],零水印模型是指从数据中提取相关属性值,不对原始数据进行修改,可保证较高的数据质量.Melkundi 等人设计出一种新型水印嵌入算法[14],该算法具有两种水印嵌入机制,可将水印嵌入到数值型和文本型的关系数据中.Rani 等人将 MapReduce 思想引入针对大型关系数据的水印嵌入,可有效提高其嵌入效率[15].

关系数据可逆水印方案由关系数据数字水印方案发展而来,目前常见的关系数据可逆水印方法主要包括直方图扩展、差值扩展、预测差值扩展等.其中,直方图扩展需首先计算数据中某些属性值的差值,并利用这些差值的首位非零数字构建直方图,然后按照一定的规则改变该非零数字的分布特性,实现水印的嵌入[16].数据恢复时,还原该数字的数据分布特性即可.该方法可追踪数据失真的程度,但难以抵御高强度的攻击.差值扩展是指从特定的元组中挑选属性值对,然后通过对该属性值对进行特定的数值变换,实现水印的嵌入和数据恢复[17].但是该方法存在以下问题:(1) 属性值对的选择使其难以应对元组修改攻击;(2) 属性值对的修改导致数据的失真程度普遍较高.针对问题(1),研究者们提出了将最优化算法和差值扩展结合使用的方法.例如,Jawad 等人在差值扩展的基础上,使用遗传算法提出了一种关系数据可逆水印方案[18].该方案在尽可能降低数据失真的同时,保证了关系数据中水印的鲁棒性.Imamoglu 等人则将萤火虫算法和差值扩展结合使用,设计出一种可逆水印方案.其中,萤火虫算法从原始数据中选择最优属性值对进行水印嵌入,以降低数据失真的程度[19].针对问题(2),研究者们提出了预测差值扩展技术.该技术通过使用预测算法获取预测值,然后从原始数据中挑选某个属性值,对其进行与差值扩展类似的数值变换,实现水印的可逆.与差值扩展相比,该技术只需修改元组中的某个属性值,对数据可用性影响较小.例如,Farfoura 等人将可识别的图像转化为比特流嵌入到数值属性的最低有效位,并通过预测误差扩展实现水印的可逆[20].Chang 等人运用支持向量回归预测(SVR)设计出关系数据可逆水印方案[21].除上述可逆水印方法外,张勇和牛夏牧运用异或的性质实现了水印的可逆[22].此方法在水印嵌入时,只需修改某些元组某个属性值的某个最低有效位,既降低了数据的失真程度,又使水印具有良好的鲁棒性.但是该方案只是将水印作为密钥用于数据恢复,不能进行水印检测,无法实现版权证明.本文所提出的分级可逆水印方案在其基础上,通过构造辅助数据,解决了水印的检测问题,并实现了水印的可逆.另外,还有一些其他类型的可逆水印方法.Franco-Contreras 等人基于圆形直方图变换提出了一种鲁棒性可逆水印方案.该方案首先构建圆形直方图,然后通过改变某些属性值的相对角位置实现水印嵌入[23].Iftikhar 等人结合遗传算法和信息论中的数据分析方法,提出了一种应用于关系数据的可逆水印方案[24].此方案首先通过遗传算法来制备最优水印串,然后通过一种可逆运算变换实现水印嵌入和数据恢复.此方案中,遗传算法的使用降低了数据的失真程度,但是计算最优水印串需要较大开销[9,24].姜传贤等人基于某种分块策略将原始数据分为若干数据块,然后将水印嵌入到数据块的小波域中,利用整数小波变换实现水印可逆[25].以上方案均可实现数据恢复,但对数据恢复的程度缺乏控制,只能将数据中的水印全部去除,导致所有者不能继续证明对数据的版权.为解决此问题,本文提出了一种分级可逆的关系数据水印方案.

2 分级可逆水印方案

本文方案定义了数据质量等级来反映数据可用性,利用异或的性质来实现水印的可逆,即 A xor B xor $B=A$,设计了用于实现分级可逆水印的分区嵌入、等级检测、水印检测以及等级提升等算法.

由图 1 可知,该方案主要包括预处理、分区嵌入、等级检测、水印检测以及等级提升这 5 部分.相关符号及含义见表 1.

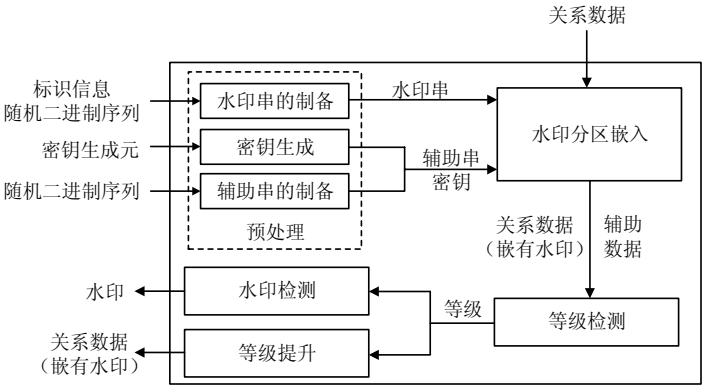

Fig.1 System structure

图 1 系统结构

Table 1 Symbols and meanings in the scheme

表 1 方案中符号及含义

符号	含义	符号	含义
β	数据所有者的身份信息	γ	分区个数
ε	属性个数	$1/\eta$	嵌入水印的比例
ξ	属性值最低有效位的位数	ω	水印串长度
μ	多数投票机制的阈值参数	τ_i	判定数据分区 i 是否有水印的阈值参数
A	属性值	A_w	嵌有水印的属性值
A_index	属性值索引	A_w_index	水印所在属性值的索引
bit_index	比特位索引	bit_w_index	水印所在比特位的索引
bit_y	属性值的第 y 位最低有效位	bit_w_y	嵌有水印的属性值的第 y 位最低有效位
bit_s	辅助数据位	bit_{ws}	bit_w_y 与 S_z 异或运算的结果
$count\_i$	水印第 i 位多数投票时的计数器	D	关系数据
D_w	嵌有水印的关系数据	D'_w	数据质量等级提升后的关系数据
D_s	辅助数据	D_s_i	数据分区 i 的辅助数据
D'_s_i	检测过程中从数据分区 i 得出的辅助数据	$Flag\_i$	数据分区 i 是否含有水印的标识
$H(p)$	以 p 为输入的哈希函数	$hash_{part}$	数据分区划分所用哈希值
$hash_{em}$	水印分区嵌入所用哈希值	hr	水印的隐藏率
k	分区划定密钥	$matchcount\_i$	数据分区 i 相关数据的匹配数
QoD	数据质量等级	r	元组
$r.key$	元组主键值	$rand_A$	随机数 A
$rand_B$	随机数 B	S	辅助串
S_z	辅助串的第 z 位	sum	嵌有水印数据的元组数
sk	分区水印嵌入密钥	sk_e	生成元（用于分区嵌入密钥生成）
sk_i	分区嵌入密钥(数据分区 i)	sk'	分区嵌入密钥子集
sk'_i	sk'中数据分区 i 的分区嵌入密钥	$table_{basic}\_i$	数据分区 i 对应的基本表
$table_{overflow}\_i$	数据分区 i 对应的溢出表	$table_{basic}\_i\_index$	$table_{basic}\_i$ 中的位置索引
$table_{overflow}\_i\_index$	$table_{overflow}\_i$ 中的位置索引	$totalcount\_i$	数据分区 i 相关数据的被检测总数
W	水印串	W_z	水印串的第 z 位
W_D	检测出的水印串	W_D_z	检测出的水印串的第 z 位

2.1 数据质量等级

为了描述水印嵌入对数据可用性的影响,方案引入了数据质量等级 QoD 的概念,如定义 1 所示.

定义 1. 数据质量等级 QoD,表示水印嵌入对数据可用性的影响.$QoD=\sigma(\sigma\in[0,\gamma-1],\sigma\in\mathbf{N})$,$\gamma$表示数据分区的个数.

由定义 1 可知,数据质量等级 QoD 具有γ种情况.其中,QoD=0,表示所有的数据分区中均嵌有水印;QoD=γ–1,表示只有 1 个数据分区嵌有水印.QoD越高,嵌有水印的数据分区越少,数据的可用性越高;反之,嵌有水印的数据分区越多,数据的可用性越低.数据使用者初期可先使用 QoD 较低的数据,如果该数据的可用性无法满足使

用者的要求,可向数据所有者申请或购买相关密钥,提升 *QoD*.*QoD* 的提升不会影响水印的有效性.

2.2 预处理

预处理阶段进行水印嵌入的准备工作,主要包括水印串和辅助串的制备以及密钥的生成等.

2.2.1 水印串和辅助串制备

水印串是数据所有者标识信息的载体,用于实现数据版权的证明.水印串通常为所有者身份信息的二进制形式[9].

水印隐藏率是水印序列长度设定的重要依据.水印隐藏率 *hr* 的计算如公式(1)所示[9].

$$hr=\frac{\omega}{sum} \tag{1}$$

其中,ω表示水印长度,*sum* 表示嵌有水印数据的元组数.在元组数相等的情况下,*hr* 越高,则ω越大,水印包含的信息量越多,每个水印位的嵌入次数越少,水印的抗攻击能力越弱;反之,则ω越小,水印包含的信息量越少,每个水印位的嵌入次数越多,水印的抗攻击能力越强.所以,水印串的长度选取应保证合适的水印隐藏率.

辅助串在等级检测、水印检测及等级提升阶段起辅助作用.辅助串的选取是随机的,与水印串等长即可.

2.2.2 密钥生成与管理

分级可逆水印的实现需要使用多个密钥,包括分区密钥 *k* 以及分区嵌入密钥 *sk*[9].为实现水印的分级可逆,方案需预设多个密钥,分区嵌入密钥 *sk* 和分区划定密钥 *k*.*k* 用于将数据划分为若干个数据分区,*sk* 用于在各数据分区中进行水印嵌入.为了提高算法的安全性和运行效率,方案设计出一种适用于分级可逆水印的密钥生成与管理机制.该机制主要包括:(1) 密钥生成;(2) 密钥分发;(3) 密钥验证;(4) 密钥存储及备份.

(1) 密钥生成

k 随机选取即可.

sk 的生成过程如下.

① 计算 $sk\_(n-1)=H(sk_e)$.其中,$H(\cdot)$为哈希函数,sk_e为随机设定的密钥生成元,$sk_(n-1)$为数据分区 *n* 的嵌入密钥.

② 计算 $sk\_(n-2)=H(sk\_(n-1))$,获得数据分区 $n-1$ 的水印嵌入密钥.

③ 重复步骤②,直至获得 $sk\_0=H(sk\_1)$,计算完毕.

由此,各数据分区的分区嵌入密钥就构成了一条密钥链.如果要对前 *i* 个数据分区进行水印嵌入或等级提升,只需知道 $sk_(i-1)$即可.另外,密钥生成元 sk_e 的选定及密钥的生成均由数据所有者实现.

(2) 密钥分发

基于本文的应用场景,数据所有者 *owner* 向数据使用者 *user* 分发等级提升所用的密钥时,使用无密钥分发协议[26].假设 *K* 为待分发密钥,*p* 为公用大素数,且 $1\leqslant K\leqslant p-2$,*owner* 和 *user* 各自选择秘密值 *a* 和 *b*,$a\geqslant 1$,$b\leqslant p-2$,且 *a*,*b* 与 $p-1$ 互素.协议内容具体如下.

① *owner* 计算 $K^a \bmod p$,并将其发送给 *user*.

② *user* 计算$(K^a)^b \bmod p$,并将其发送给 *owner*.

③ *owner* 计算$(K^{ab})^{-a} \bmod p=K^b \bmod p$,并将其发送给 *user*.

④ *user* 计算$(K^b)^{-b} \bmod p=K \bmod p$,此密钥分发完成.

(3) 密钥验证

为防止密钥在传输过程中遭到破坏,导致数据质量等级提升失败,数据使用者需对密钥进行验证.

假设 *rand* 为随机值,*K* 为分发的密钥,*rand_hash*=*H*(*rand*,*K*).密钥分发完毕之后,*owner* 还需将 *rand* 和 *rand_hash* 发送给 *user*.其后,*user* 计算 *H*(*rand*,*K*),如果该值与 *rand_hash* 相同,证明密钥无误,可用于数据质量等级提升;否则,重新进行密钥分发.

(4) 密钥存储及备份

密钥存储采用物理与加密结合的方式.密钥存储时,首先使用 RSA 加密算法对密钥进行加密[27],其后,将密

文和密钥加密密钥分别存储于专用的 USB 磁盘.

密钥备份采用 Shamir 的(k,n)门限秘密共享方案($k\leqslant n$)[28].数据所有者可使用该方案将密钥分为 n 份子密钥,并分别交由不同的管理人员保管或者在不同的存储位置保存.后期进行密钥恢复时,仅需 k 份密钥即可.

其中,密钥生成在预处理阶段完成.要注意的是,本方案所涉及密钥主要用于水印嵌入及检测等相关操作,不需要频繁的密钥更换.因此与一般的数据加密方案相比,本方案中密钥管理相对简单且容易实现,不考虑密钥更新和密钥销毁的问题.

2.3 水印分区嵌入

水印分区嵌入阶段的任务是将可声明数据版权的水印嵌入到数据当中,同时预设数据质量等级,以便后期进行水印检测和数据质量等级操作.该阶段的主要工作包括数据分区的划分、辅助数据的计算和存储以及水印的嵌入等.水印分区嵌入算法如算法 1 所示.

算法 1. 水印分区嵌入算法.

Input: k,sk,D,W,S.

Output: D_s,D_w.

1. **for each** tuple $r\in D$ **do**
2. $hash_{part}=H(r.key,k)$;
3. $i=hash_{part}$ mod γ;
4. $hash_{em}=H(r.key,sk\_i)$;
5. **if** ($hash_{em}$ mod η equals 0) {
6. $A\_index\ x=hash_{em}$ mod ε;
7. **if** (A_x is *NULL*)
8. *continue*(·);
9. $bit\_index\ y=hash_{em}$ mod ξ;
10. $WS\_indexz=hash_{em}$ mod ω;
11. $bit_s=bit\_y$ xor S_z;
12. *Store*(D_S_i,bit_s);
13. $bit_w=bit\_y$ xor W_z;
14. *Update*(bit_y,bit_w);}
15. **else**
16. next tuple;
17. **Return** D_w,D_s;

水印分区嵌入算法执行前需设定数据分区的个数γ,预设数据质量等级.在数值上,数据分区的个数等于数据质量等级数.算法 1 给出了水印分区嵌入的主要步骤:对于任意元组 r,首先计算分区哈希值 $hash_{part}$,确定 r 所在数据分区.数据分区的确定由 $hash_{part}$ 和数据分区数γ决定.其中,$hash_{part}$ 的计算以主键 $r.key$ 和分区密钥 k 为输入.相同的输出值 i 归为同一数据分区.然后,计算嵌入哈希值 $hash_{em}$,依次选择待嵌入水印的元组、属性值、比特位.其中,$hash_{em}$ 的计算以 $r.key$ 和分区嵌入密钥 sk_i,ε,ξ,ω分别表示数据属性个数、属性值最低有效位的位数、水印串和辅助串的长度.另外,算法在执行过程中遇到空值,则跳入下一循环,处理下一元组.其后,计算并存储辅助数据 bit_s.其中,*Store*(D_s_i,bit_s)表示将辅助数据 bit_s 存入数据分区 i 的辅助数据存储结构 D_s_i,其具体存储过程见函数 1.bit_s 存储完毕之后,执行水印嵌入操作.需要说明的是,辅助数据的计算和水印的嵌入均基于异或运算 xor 实现.最后,输出嵌入水印的数据 D_w 和所有的辅助数据 D_s,水印分区嵌入完成.

函数 1. *Store*(D_s_i,bit_s).

1. $table_{basic}\_i\_indexk=hash_{em}$ mod m;
2. **if** ($table_{basic}\_i$[k] is *NULL*)

3. $table_{basic\_}i[k]=bit_s$;
4. **else if** ($table_{basic\_}i[k]$ euqals bit_s)
5. *continue*(·);
6. **else if** ($table_{basic\_}i[k]$!euqals bit_s)
7. *put*($hash_{em}$,$table_{overflow\_}i$);

辅助数据存储时,采用哈希表为主的存储结构,且以数据分区为单位进行存储,各数据分区 i 均具有相应的辅助数据存储结构 D_s_i.D_s_i 由两张表构成:一张基本表 $table_{basic\_}i$,一张溢出表 $table_{overflow\_}i$.其中,$table_{basic\_}i$ 为哈希表,用于存储大部分辅助数据;$table_{overflow\_}i$ 为普通的关系数据表,用于存储发生哈希冲突且取值相反的辅助数据对应的嵌入哈希值 $hash_{em}$.根据函数 1,bit_s 的具体存储过程:首先计算 bit_s 在 $table_{basic\_}i$ 中的存储位置 k,其中,m 表示 $table_{basic\_}i$ 的规模.如果 $table_{basic_}i[k]$为空,将 bit_s 存入 $table_{basic_}i[k]$即可;否则判断 $table_{basic_}i[k]$与 bit_s 是否相同.如果相同,不做处理,函数 1 运行完毕;如果不同,将 bit_s 对应的 $hash_{em}$ 存入 $table_{overflow\_}i$,函数 1 运行完毕.要注意的是,待所有的辅助数据存储完毕后,需为 $table_{overflow\_}i$ 建立数据库索引,以提高 $table_{overflow}$ 的查询效率.另外,$table_{overflow\_}i$ 存储的是 $hash_{em}$.相比于存储主键值本身的方式,其占用空间更少,而且对于具有非数值型主键的数据,以及具有多字段组合主键的数据来说,可以采用统一的处理方式,其效率更高,更具实用性.

为降低辅助数据存储时的冲突率,进一步提高辅助数据的存取效率,本方案对辅助数据存储结构中基本表的规模进行了实验分析.具体的实验细节见第 4.1 节.

2.4 数据质量等级检测

数据质量等级的检测是实现分级可逆水印的一个重要环节,其本质是检测嵌有水印的数据分区.水印检测和数据质量等级提升都需要提前获知当前的数据质量等级,确定水印所在的数据分区.具体流程见算法 2.

算法 2. 等级检测算法.

Input: D_s,D_w,k,sk,S.

Output: *Flag*.

1. **for** i=0 to γ−1 do
2. $totalcount\_i$=$matchcount\_i$=0;
3. **for each** tuple $r\in D_w$ **do**
4. $hash_{part}=H(r.key,k)$;
5. $i=hash_{part}$ mod γ;
6. $hash_{em}=H(r.key,sk\_i)$;
7. **if** ($hash_{em}$ mod η equals 0) {
8. $A\_index\ x=hash_{em}$ mod ε;
9. **if** (A_x is *NULL*)
10. *continue*(·);
11. $bit\_index\ y=hash_{em}$ mod ξ;
12. $WS\_index\ z=hash_{em}$ mod ω;
13. $bit_s=bit\_y$ xor S_z;
14. $bit\_temp=Get(r.key,D_s\_i)$;
15. **if** (bit_s equals bit_temp)
16. $matchcount\_i$++;
17. $totalcount\_i$++;}
18. **else**
19. next tuple;
20. $\tau\_i=threshold(totalcount\_i,\alpha)$;

21. **for** i=0 to γ−1 do
22. **if** ($matchcount\_i \geqslant \tau\_i$)
23. $Flag\_i$=0;
24. **else**
25. $Flag\_i$=1;
26. **Return** *Flag*;

由算法 2 可知,等级检测算法和水印分区嵌入算法在数据分区的确定、水印嵌入位置的选择、辅助数据的计算方面相同.辅助数据 bit_s 计算完毕之后,从 D_s_i 中取出之前存储的辅助数据 *bit_temp*,并比较 bit_s 和 *bit_temp*:如果二者相同,匹配总数 *matchcount_i* 加 1,*totalcount_i* 加 1;否则,仅 *totalcount_i* 加 1.其中,*bit_temp* 的提取过程具体见函数 2.

函数 2. *Get*(*r.key*,D_s_i).

1. $table_{basic}\_i\_indexk=hash_{em}$ mod m;
2. **if** ($table_{basic}\_i[k]$ is *NULL*)
3. **Return** −1;
4. **else** {
5. **if** ($hash_{em}.exixtedIn(table_{overflow}\_i)$)
6. Return $table_{basic}\_i[k].reverse(\cdot)$;
7. **else**
8. **Return** $table_{basic}\_i[k]$;}

所有的辅助数据统计完毕之后,依次判定所有的数据分区中是否包含水印.首先选择判定参数 τ_i,如果 $matchcount\_i \geqslant \tau\_i$,*Flag_i* 置 0,表示数据分区 *i* 中不包含水印;如果 $matchcount\_i < \tau\_i$,*Flag_i* 置 1,表示数据分区 *i* 中包含水印.最后返回 *Flag*,等级检测完成.其中,

- τ_i 表示判定数据分区 *i* 是否包含水印的参数.
- $threshold(totalcount_i,\alpha)$的返回值是满足 $\sum_{\tau}^{totalcount\_i} \frac{totalcount\_i!}{totalcount\_i!(totalcount\_i-\tau)!}(1/2)^{totalcount\_i} < \alpha$ 的参数 τ,其具体分析见第 3 节.

根据函数 2,从 D_s_i 提取元组 *r* 的辅助数据的具体过程是:首先计算 *r* 的辅助数据在 $table_{basic}\_i$ 中的存储位置 *k*.然后查询 $table_{basic}_i[k]$是否为空.如果 $table_{basic}\_i$ 为空,返回−1;如果不为空,查询 $table_{overflow}\_i$ 是否存在 $hash_{em}$.如果存在,返回 $table_{basic}_i[k]$的相反值;否则,返回 $table_{basic}_i[k]$的值.

2.5 水印检测

水印检测是实现关系数据版权保护的重要环节.检测出的水印串在逻辑异或解密之后,可还原出数据所有者的标识信息,实现版权验证.

由水印分区嵌入算法可知,为了提高水印的抗攻击能力,各水印位被重复多次嵌入到原始数据中.如果嵌有水印的数据遭受攻击,则受攻击元组可能会干扰水印检测,导致部分检测结果出错,所以需要一个少数服从多数的投票过程来消除受攻击元组的干扰,以提高水印检测准确率.故方案采用了多数投票机制,以表 2 为例.

Table 2 Majority vote

表 2 多数投票

检测结果 1	0	1	1	1	0
检测结果 2	1	1	1	1	0
检测结果 3	0	1	1	1	1
投票结果	0	1	1	1	0

故水印检测的最终结果为 01110.

水印检测算法如算法 3 所示.

算法 3. 水印检测算法.

Input: D_w,D_s,k,sk,S,$Flag$.

Output: W_D.

```
1.  for i=0 to ω−1 do
2.      count_i=totalcount_i=0;
3.  for each tuple r∈D_W do
4.      hash_part=H(r.key,k);
5.      i=hash_part mod γ;
6.      hash_em=H(r.key,sk_i);
7.      if (hash_em mod η equals 0) {
8.          if (Flag_i=1) {
9.              A_w_index x=hash_em mod ε;
10.             if (A_x is NULL)
11.                 continue(·);
12.             bit_w_index y=hash_em mod ξ;
13.             WS_index z=hash_em mod ω;
14.             bit_s=Get(r.key,D_s_i);
15.             if (bit_s equals −1)
16.                 continue(·);
17.             else {
18.                 bit_ws=bit_w_y xor S_z;
19.                 W_D_z=bit_s xor bit_ws;
20.                 if (W_D_z equals 1)
21.                     count_z++;
22.                 else
23.                     count_z−−;
24.                 totalcount_z++;}
25.     else
26.         next tuple;}}
27. for i=0 to ω−1 do
28.     if (|count_i|≥totalcount_i×μ){
29.         if (count_i>0)
30.             W_D_i=1;
31.         else
32.             W_D_i=0;}
33.     else
34.         W_D_i=−1;
35. Return W_D;
```

如算法 3 所述,水印检测算法的执行过程如下.

- 首先,初始化计数器.其中,*count_i* 表示对水印的第 *i* 位进行多数投票所用的计数器,*totalcount_i* 表示水印第 *i* 位的总检测次数.

- 然后,确定元组所在数据分区、选择水印嵌入位置、提取辅助数据,此部分与等级检测算法相同.辅助数据 bit_s 提取完毕之后,需判定其是否等于–1:如果等于–1,表示 bit_s 无效,并跳入下一循环,检测下一元组;否则,计算包含水印元组的辅助数据 bit_{ws},执行水印检测操作.
- 接着,执行多数投票.如果 W_D_z 等于 1,$count\_i$ 加 1;反之,$count\_i$ 减 1.
- 所有元组检测完毕之后,统计多数投票结果.先判断|$count\_i$|是否大于或等于 $totalcount\_i$ 与参数μ的乘积:如果满足条件,表示投票结果有效,继续判断出水印位即可;否则,令 W_D_i 为–1,表示此水印位的多数投票无效.其中,参数μ是个比例阈值,用来提高多数投票机制的有效性.比如,$totalcount\_i$=100,μ等于 0.02,则|$count\_i$|等于 0 或 1 均不能作为水印位判定的依据.

μ的引入增加了水印检测算法的可靠性,但如果将μ设置的过高,则会导致大量的水印位检测无效.所以,用户根据实际应用的要求对μ进行设定.

2.6 数据质量等级提升

数据质量等级提升的本质是去除某些数据分区中的水印,控制嵌有水印的数据分区数.假如数据使用者不满足当前数据的可用性,可向数据所有者申请或购买密钥进行数据质量等级提升,以提高数据可用性.另外,数据质量等级提升的程度不受限制,既可以逐级提升,也可以越级提升.等级提升算法如算法 4 所示.

算法 4. 等级提升算法.

Input: D_w,D_s,k,sk',$Flag$.

Output: D'_w.

1. **for each** tuple $r \in D_W$ **do**
2. $hash_{part}=H(r.key,k)$;
3. $i=hash_{part}$ mod γ;
4. $hash_{em}=H(r.key,sk'\_i)$;
5. **if** ($hash_{em}$ mod η equals 0) {
6. **if** ($Flag\_i$=1) {
7. $A_w\_index\ x=hash_{em}$ mod ε;
8. **if** (A_x is $NULL$)
9. $continue(\cdot)$;
10. $bit_w\_index\ y=hash_{em}$ mod ξ;
11. $WS\_index\ z=hash_{em}$ mod ω;
12. $bit_s=Get(r.key,D_s\_i)$;
13. **if** (bit_s equals –1){
14. $continue(\cdot)$; }
15. **else** {
16. $bit_{ws}=bit_w\_y$ xor S_z;
17. $W_D\_z=bit_s$ xor bit_{ws};
18. $bit\_y=bit_w\_y$ xor W_D_z;
19. $Update(bit_w\_y,bit\_y)$; }}
20. **else**
21. next tuple;
22. **Return** D'_w;

由算法 4 可知,等级提升算法与水印检测算法在确定元组的数据分区、选择水印嵌入位置、辅助数据的提取以及具体水印位的检测均相同.检测出当前元组中嵌入的水印位 W_D_z 后,计算原始比特位 bit_y,并执行更新操作.最后返回等级提升后的关系数据 D'_w,等级提升完成.需要注意的是,输入的密钥 sk'只包含当前数据质量等

级提升所需的分区嵌入密钥,不会暴露剩余数据分区中水印的嵌入位置.

3 概率分析与参数选择

等级检测算法的基本原理是:从待检测元组中检测出匹配信息,然后将匹配总数与相应参数进行比较,根据比较结果,得出检测结果.理论上,等级检测算法中的参数$\tau_i(0\leqslant i\leqslant\gamma-1)$应等于其待检测元组的总数.但在实际的应用中,水印可能会遭受攻击,所以$\tau_i(0\leqslant i\leqslant\gamma-1)$一般小于或等于理论值.本节对$\tau_i(0\leqslant i\leqslant\gamma-1)$进行了概率分析,为其在实际应用中的选择提供参考.考虑$\tau_i(0\leqslant i\leqslant\gamma-1)$的选择方式相同,本章的后续部分使用$\tau$来泛指参数$\tau_i(0\leqslant i\leqslant\gamma-1)$.

3.1 概率分析

假设使用等级检测算法对不包含水印的数据进行检测,则从待检测元组中检测到匹配信息的概率约为 1/2.我们可以将从 n 个元组中检测到τ个匹配信息的过程看作是τ次成功的 n 重 Bernoulli 实验,每次实验的成功概率为 1/2,其概率表示如公式(2)所示.

$$P[\tau\ matches]=b(\tau;n,1/2)=\frac{n!}{\tau!(n-\tau)!}(1/2)^n \tag{2}$$

令$\varphi=\tau/n$,表示检测到匹配信息的元组比例,则φ的概率分布 $P(\varphi)$满足二项分布,其分布曲线如图 2 所示.

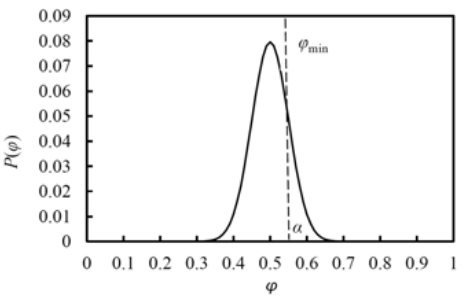

Fig.2 Probability distribution of φ with the original data

图 2 原始数据中φ的概率分布

假设$\varphi_{\min}=\tau_{\min}/n$,则由图 1 可知,$\varphi$位于区间$[\varphi_{\min},1]$的概率 $P\{\varphi_{\min}\leqslant\varphi\leqslant1\}=\alpha$,而 $P\{\varphi_{\min}\leqslant\varphi\leqslant1\}$的概率表示如公式(3)所示.

$$P\{\varphi_{\min}\leqslant\varphi\leqslant1\}=\sum_{\tau=\tau_{\min}}^{n}b(\tau;n,1/2)=\sum_{\tau=\tau_{\min}}^{n}\frac{n!}{\tau!(n-\tau)!}(1/2)^n \tag{3}$$

所以,根据概率论中的“小概率原理”,当α很小时,φ几乎不可能落在区间$[\varphi_{\min},1]$内.也就是说,从不包含水印的数据中至少检测出$\tau_{\min}$ 个匹配信息几乎是不可能发生的.而水印嵌入的目的即改变这种概率分布,使得从嵌有水印的数据中检测出超过$\tau_{\min}$ 的匹配信息成为非偶然事件.所以,τ是否落在区间$[\tau_{\min},n]$即可作为判定数据中是否包含水印的依据.

3.2 参数选择

由第 3.1 节可知,τ的取值取决于 n 和α.其中,n 由关系数据的元组总数和水印嵌入比例决定,无法随意修改.所以,此处主要讨论α的取值对τ的影响.

假设 ar 表示某包含水印的关系数据允许被修改的元组比例,其计算公式如公式(4)所示.

$$ar=\frac{n-\tau_{\min}}{n}=1-\varphi_{\min} \tag{4}$$

改变α,ar 的变化如图 3 所示.

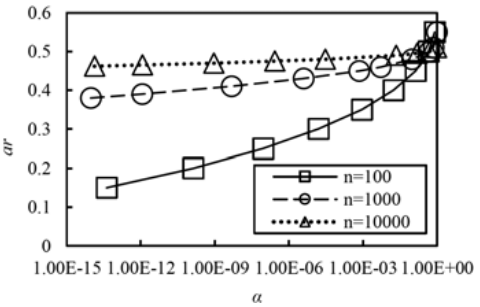

Fig.3 Relation between *ar* and α

图 3 *ar* 和α的关系

由图 3 可知,对于不同的 *n*,*ar* 随α的增大而增大.也就是说,显著度参数越大,允许被修改的元组比例越大,水印的抗攻击能力越强.但是,*ar* 的增大,势必会导致$\varphi_{\min}$的减小,即$\tau_{\min}$的减小.基于之前的概率分析,$\tau_{\min}$的减小将会导致检测结果误判率的升高.所以,在水印嵌入数据 *n* 一定的情况下,应该综合考虑水印的抗攻击能力以及误判率来选择相关参数.

4 实验分析

本节对提出的可逆水印方案进行了实验验证,实验环境为:CPU 主频 2.70GHz,内存 4GB,操作系统 Windows10,数据库 MySQL 5.7,开发环境 eclipse 4.5.2[9].数据采用 1GB 的 TPC-H 数据集[29].具体来说,选择并改造 partsupp 数据表,最后设定属性数为 5,元组数为 200 000[9].为验证方案的可行性和鲁棒性,实验包括算法的计算性能实验和水印的抗攻击能力实验.实验结果均为 5 次实验的平均值.

4.1 算法的计算性能实验

4.1.1 水印分区嵌入算法的计算性能实验

实验对选取出的属性值的小数部分进行水印嵌入,设定γ=20,ξ=5,1/η=1/4.假设数据的元组数为 *n*,我们将水印分区嵌入算法与普通的读取 *n* 条元组并写入 *n*/4 条元组的时间进行对比.设定 *n*=5000,10000,20000,50000,100000,200000,实验结果如图 4 所示.

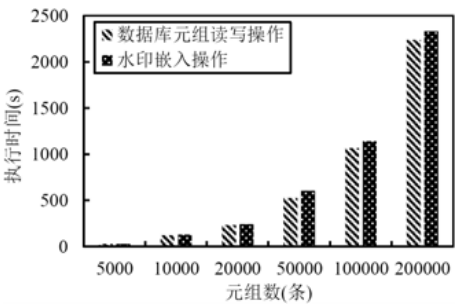

Fig.4 Time comparison between watermark embedding and database reading and writing

图 4 水印分区嵌入与数据库读写操作时间对比

.在执行时间方面,该算法较普通的数据库元组读写操作增长了约 6.4%.增加的开销源于划定数据分区、选取嵌入水印位以及计算存储辅助数据等.

4.1.2 等级检测算法的计算性能实验

考虑等级检测算法只需读取辅助数据,而不涉及数据写入操作.故该实验将等级检测算法与数据库读取 *n* 条元组的时间进行对比,如图 5 和图 6 所示.

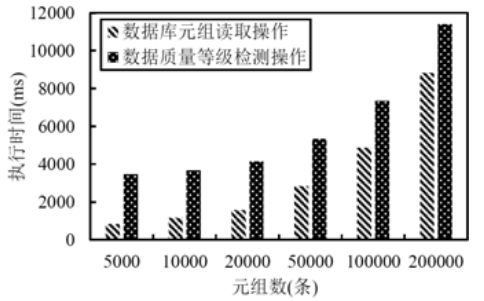

Fig.5 Time comparison between grade detection and database reading

图 5 等级检测与数据库读取操作时间对比

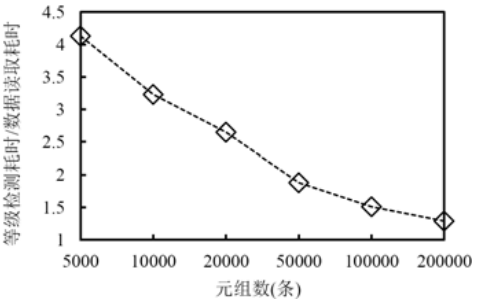

Fig.6 Change of grade detection time/ database reading time

图 6 等级检测耗时/数据库读取操作耗时变化

经过对实验结果的分析,n=5000 时,等级检测算法的执行时间约为普通的数据库元组读取操作的 4 倍.但随着数据规模 n 的增大,等级检测算法与普通的数据库元组读取操作的时间比值呈下降趋势.当 n=200000 时,该比值已经下降到 1.29,如图 6 所示.这是因为等级检测算法在执行初期需要读取水印嵌入期间保存下来的辅助数据,并且在执行后期要进行辅助数据的比对,这两部分时间受数据规模的影响不大,但从整体上增加了算法的时间开销.

4.1.3 水印检测算法的计算性能实验

水印检测算法同样只涉及数据库的元组读取操作.故该算法只与读取 n 条数据库元组的时间对比,如图 7 和图 8 所示.

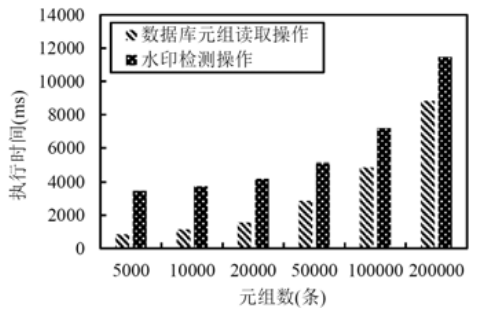

Fig.7 Time comparison between watermark detection and database reading

图 7 水印检测与数据库读取操作时间对比

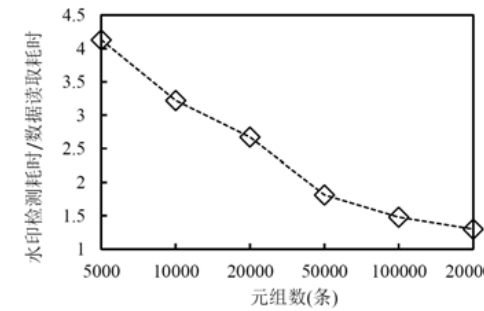

Fig.8 Change of watermark detection time/ database reading time

图 8 水印检测耗时/数据库读取操作耗时变化

由图 7 和图 8 可知,水印检测算法在计算性能上与等级检测算法具有相似的性质,都是算法运行时间与读取数据库元组时间的差距逐渐缩小.其原因也是受辅助数据读取的影响.

4.1.4 等级提升算法的计算性能实验

实验是对嵌有水印的关系数据执行数据恢复操作,以提升数据质量等级.实验将算法时间与读取 n 条数据库元组写入 $n/4$ 条数据库元组对比,结果如图 9 所示.

图 9 中,在执行时间上,与普通的数据库元组读写操作相比,等级提升算法相对于普通的读写数据库元组而言,增加了约 6.0%.增加的时间主要源于恢复原始数据、划定数据分区等.

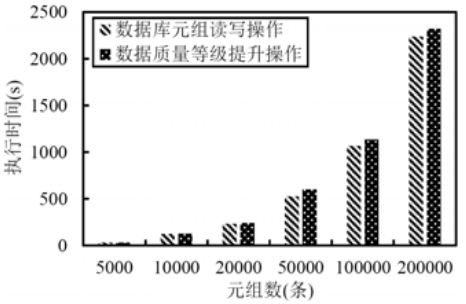

Fig.9 Time comparison between grade enhancement and database reading and writing

图 9 等级提升与数据库读写操作时间对比

4.2 水印的抗攻击能力实验

水印的抗攻击能力实验是模拟攻击者对嵌有水印的关系数据进行各种攻击,验证水印抵御各种攻击的能力.攻击形式主要包括:(1) 元组删除攻击;(2) 元组添加攻击;(3) 元组修改攻击.实验从水印检测准确率 *wdr* 和数据恢复准确率 *drr* 对水印的抗攻击能力和数据的恢复能力进行评估,*wdr* 和 *drr* 的计算公式如公式(5)和公式(6)所示.

$$wdr = \frac{bits_{WD}}{\omega} \tag{5}$$

$$drr = \frac{tuples_{DR}}{tuples_{D}} \tag{6}$$

其中,bit_{WD} 表示检测出的正确的水印位数,ω表示水印串长度,$tuples_{DR}$ 表示数据恢复正确的元组数,$tuples_{D}$ 表示关系数据中的元组数.

实验参数设置如下:数据中元组数 200 000 条,γ=20,ξ=5,1/η=1/4,ω=53,hr=0.0053,μ=0.02.

同时,实验将本方案与 DEW[17]、GADEW[18]、PEEW[20]以及 RRW[24]的水印抗攻击能力和数据恢复能力做了对比.实验随机选取数据分区,并以数据分区为单位进行水印检测及数据恢复,根据实验参数设置,每个数据分区约有 10 000 条元组.

4.2.1 元组删除攻击实验

元组删除攻击是指从包含水印的关系数据中去除一定比例的元组,以减少嵌有水印的元组的数目,达到破坏水印的目的[9].此种攻击在破坏水印的同时,会牺牲掉部分未嵌入水印的元组,导致数据可用性的下降.因此在实际的应用场景中,元组删除攻击的规模往往是有限的.本实验为充分地证明方案的鲁棒性,将元组删除规模的上限设置为 90%.实验结果如图 10 和图 11 所示.

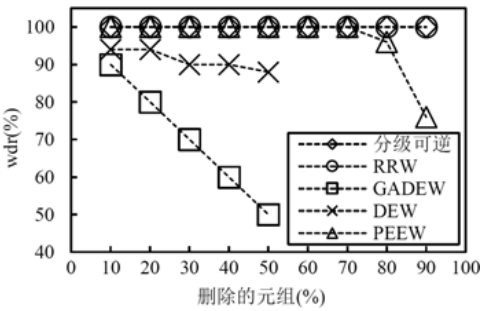

Fig.10 Watermark detection accuracy after tuples deletion attacks

图 10 水印检测的准确率(元组删除攻击)

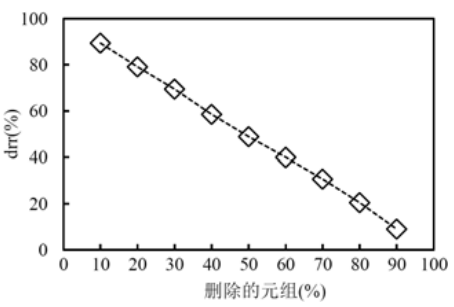

Fig.11 Data recovery accuracy after tuples deletion attacks

图 11 数据恢复的准确率(元组删除攻击)

由图 10 可知,该方案的 *wdr* 明显高于 GADEW,DEW.当元组删除比例大于 70%之后,PEEW 的 *wdr* 开始下降,而 RRW 和该方案则一直保持在 100%.这说明该方案抵御元组删除攻击的能力优于 GADEW、DEW 及 PEEW,不逊色于 RRW.

由图 11 可以看出,方案的 *drr* 随删除的元组规模的增大呈线性递减趋势.该实验结果的出现,主要有以下两个原因.

(1) 元组删除攻击将部分嵌有水印的元组删除,无法进行数据恢复,而剩余的嵌有水印的元组能够全部恢复为原始数据.随着删除的元组比例不断增大,*drr* 势必呈现出递减趋势.

(2) 由于删除的元组比例与数据恢复的准确率之和近似于 1,故 *drr* 呈线性递减趋势.

4.2.2 元组添加攻击实验

元组添加攻击是指向包含水印的关系数据中添加一定比例的元组,这不会对原有的数据产生影响,但会扰乱水印检测,致使水印检测错误[9].为充分说明水印抵御元组添加攻击的能力,实验将元组添加比例的上限设置为 100%,实验结果如图 12 和图 13 所示.

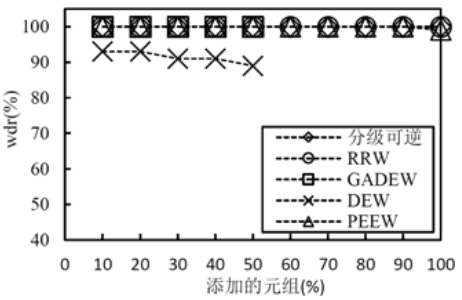

Fig.12 Watermark detection accuracy after tuples adding attacks

图 12 水印检测的准确率(元组添加攻击)

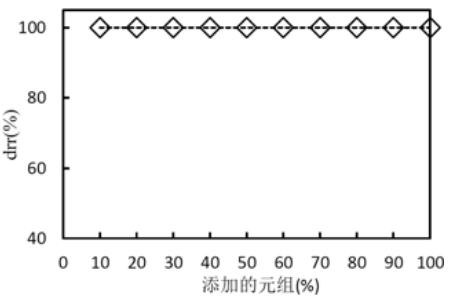

Fig.13 Data recovery accuracy after tuples adding attacks

图 13 数据恢复的准确率(元组添加攻击)

从图 12 中可知,当元组的比例不大于 50%时,该方案的 *wdr* 明显高于 DEW,与其余方案持平.当元组的比例大于 50%时,GADEW 的 *wdr* 降到 40%以下,而该方案和 RRW 始终保持在 100%.这说明该方案可以有效地抵御元组添加攻击.

从图 13 中可知,当添加元组的比例不高于 100%时,*drr* 保持在 100%.这是由于水印嵌入期间,元组、属性值及相应数位的选取均由以元组主键值为输入的哈希函数决定,元组添加攻击不会改变相应的哈希值,因此数据恢复时,可准确定位到嵌有水印的数位.

另外,数据恢复时可能会修改元组添加攻击产生的新元组,此处不做考虑.

4.2.3 元组修改攻击实验

元组修改攻击是指在包含水印的关系数据中修改一定比例的元组,致使水印检测出错,达到破坏水印的目的[9].此种攻击往往造成数据可用性的下降,甚至数据无法使用.为充分验证水印在抵御元组修改攻击方面的能力,本实验选择最为严格的元组修改方式,即修改元组时,修改元组中的全部属性,这势必会修改水印所在的属性值.由此得到实验结果如图 14 和图 15 所示.

由图 14 可知,元组修改的比例在不高于 80%的情况下,该方案和 RRW 的 *wdr* 均保持在 100%,高于其他方案.当元组修改的比例达到 90%时,该方案的 *wdr* 为 95.02%,低于 RRW,但高于其他方案.这是由于在水印检测时,未受修改攻击的元组被遭受修改攻击的元组干扰,致使水印在检测时出现错误.但是在实际应用场景中,往往不会遭受如此严格且如此高比例的元组修改攻击.所以,该方案足以抵御大多数的元组修改攻击.

该实验 *drr* 与元组删除攻击实验显示出类似的线性递减趋势,如图 15 所示.这是因为此种元组修改攻击较为严格,遭受攻击的元组恢复为原始数据的可能性较小.线性趋势的形成原因与元组删除攻击相同.

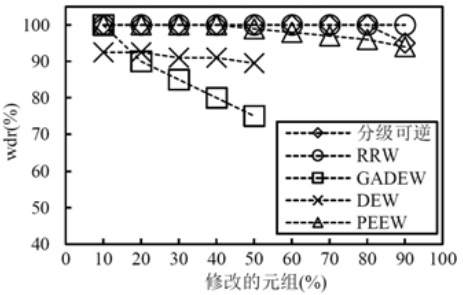

Fig.14 Watermark detection accuracy after tuples alteration attacks

图 14 水印检测的准确率(元组修改攻击)

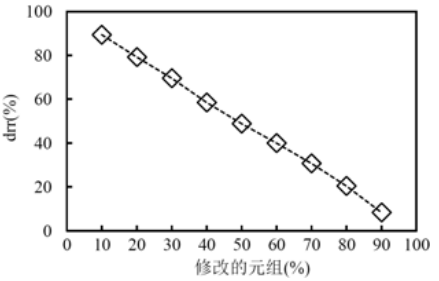

Fig.15 Data recovery accuracy after tuples alteration attacks

图 15 数据恢复的准确率(元组修改攻击)

5 结束语

传统的关系数据数字水印方案会永久性地损害数据的可用性,使得可用性难以满足使用者的要求.现有的关系数据可逆水印方案可将数据中的水印全部去除,实现数据恢复.但是数据恢复后,其版权将无法得到保护.本文针对上述问题,提出了一种分级可逆的关系数据水印方案,定义了数据质量等级来反映数据的可用性,设计了可实现分级可逆水印的分区嵌入、等级检测、水印检测以及等级提升等算法,可以对任何数据质量等级的数据进行等级提升.同时,对于任意数据质量等级的关系数据,均可通过其中的水印证明数据的版权.实验分析表明,方案中算法具有较高的执行效率,可满足绝大多数应用场景的要求.水印具有良好的抗攻击能力,足以应对各类攻击.

References:

[1] Naimi AI, Westreich DJ. Big data: A revolution that will transform how we live, work, and think. Mathematics & Computer Education, 2014,47(17):181−183.

[2] Katzenbeisser S, Petitcolas FA. Information Hiding Techniques for Steganography and Digital Watermarking. Artech House, Inc., 2000.

[3] Alattar AM. Reversible watermark using the difference expansion of a generalized integer transform. IEEE Trans. on Image Processing, 2004,13(8):1147−1156.

[4] An L, Gao X, Li X, Tao D, Deng C, Li J. Robust reversible watermarking via clustering and enhanced pixel-wise masking. IEEE Trans. on Image Processing, 2012,21(8):3598−3611.

[5] Cox IJ, Kilian J, Leighton T, Shamoon T. Secure spread spectrum watermarking for multimedia. IEEE Trans. on Image Processing, 1997,6(12):1673−1687.

[6] Hartung F, Kutter M. Multimedia watermarking techniques. Proc. of the IEEE, 1999,87(7):1079−1107.

[7] Agrawal R, Kiernan J. Watermarking relational databases. In: Proc. of the Int'l Conf. on Very Large Databases. 2002. 155−166.

[8] Sion R, Atallah M, Prabhakar S. Rights protection for categorical data. IEEE Trans. on Knowledge and Data Engineering, 2005, 17(7):912−926.

[9] Hou RT, Xian HQ, Liu HY, Gao Y, Zhang Y. Multi-verifiable Reversible Watermarking Scheme for Relational Data. Journal of Cryptologic Research, 2019,6(1):37−49 (in Chinese with English abstract).

[10] Niu XM, Zhao L, Huang WJ, Zhang H. Watermarking relational databases for ownership protection. Acta Electronica Sinica, 2004, 31(12A):2050−2053 (in Chinese with English abstract).

[11] Zhang ZH, Jin XM, Wang JM, Li DY. Watermarking relational database using image. In: Proc. of the Int'l Conf. on Machine Learning and Cybernetics, Vol.3. IEEE, 2004. 1739−1744.

[12] Xiang Y, Li JY, Pan JF. Database watermarking algorithm based on virtual primary key. Journal of Computer Research and Development, 2009,46(Suppl.):66−70 (in Chinese with English abstract).

[13] Zhou G, Wu KM. Research on zero-watermarking model of relational databases based on improved C4.5 algorithm. Computer Applications and Software, 2015,32(1):64−67 (in Chinese with English abstract).

[14] Melkundi S, Chandankhede C. A robust technique for relational database watermarking and verification. In: Proc. of the Int'l Conf. on Communication, Information & Computing Technology. IEEE, 2015. 1−7.

[15] Rani S, Koshley DK, Halder R. Adapting MapReduce for efficient watermarking of large relational dataset. In: Proc. of the 2017 IEEE Trustcom/BigDataSE/ICESS. 2017. 729−736.
[16] Zhang Y, Yang B, Niu XM. Reversible watermarking for relational database authentication. Journal of Computers, 2006,17(2): 59−65.
[17] Gupta G, Pieprzyk J. Reversible and blind database watermarking using difference expansion. In: Proc. of the DBLP. 2008. 1−6.
[18] Jawad K, Khan A. Genetic algorithm and difference expansion based reversible watermarking for relational databases. Journal of Systems & Software, 2013,86(11):2742−2753.
[19] Imamoglu MB, Ulutas M, Ulutas G. A new reversible database watermarking approach with firefly optimization algorithm. Mathematical Problems in Engineering, 2017,2017(2):1−14.
[20] Farfoura ME, Horng SJ. A novel blind reversible method for watermarking relational databases. In: Proc. of the Int'l Symp. on Parallel and Distributed Processing with Applications. IEEE Computer Society, 2010. 563−569.
[21] Chang JN, Wu HC. Reversible fragile database watermarking technology using difference expansion based on SVR prediction. In: Proc. of the Int'l Symp. on Computer, Consumer and Control. IEEE, 2012. 690−693.
[22] Zhang Y, Niu XM. Reversible watermark technique for relational databases. Acta Electronica Sinica, 2006,34(12A):2425−2428 (in Chinese with English abstract).
[23] Franco-Contreras J, Coatrieux G, Cuppens F, Cuppens-Boulahia N, Roux C. Robust lossless watermarking of relational databases based on circular histogram modulation. IEEE Trans. on Information Forensics & Security, 2014,9(3):397−410.
[24] Iftikhar S, Kamran M, Anwar Z. RRW—A robust and reversible watermarking technique for relational data. IEEE Trans. on Knowledge & Data Engineering, 2015,27(4):1132−1145.
[25] Jiang CX, Cheng XH, Xu XL, Li Z. Reversible database watermark based on integer wavelet transform. Journal of Guilin University of Technology, 2017,37(1):191−195 (in Chinese with English abstract).
[26] Feng DG. Security Protocols: Theory and Practice. Beijing: Tsinghua University Press, 2011 (in Chinese).
[27] Hu Y. Research and implementation of RSA algorithm [MS. Thesis]. Beijing: Beijing University of Posts and Telecommunications, 2010 (in Chinese with English abstract).
[28] Rong HG, Mo JX, Chang BG, Sun G, Long F. Key distribution and recovery algorithm based on Shamir's secret sharing. Journal on Communications, 2015,36(3):60−69 (in Chinese with English abstract).
[29] TPC-H. 2018. http://www.tpc.org/tpch/

附中文参考文献:

[9] 侯瑞涛.咸鹤群.刘红燕.高原.张艺.可多次验证的关系数据可逆水印方案.密码学报,2019,6(1):37−49.
[10] 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,2004,31(12A):2050−2053.
[12] 向玥,李军义,潘季芳.一种基于虚拟主键的数据库水印算法.计算机研究与发展,2009,46(Suppl.):66−70.
[13] 周钢,吴克明.基于改进型 C4.5 算法的关系数据库零水印模型研究.计算机应用与软件,2015,32(1):64−67.
[22] 张勇,牛夏牧.一种用于关系数据库的可逆水印技术.电子学报,2006,34(12A):2425−2428.
[25] 姜传贤,程小辉,许鑫磊,李智.基于整数小波变换的可逆数据库水印.桂林理工大学学报,2017,37(1):191−195.
[26] 冯登国.安全协议:理论与实践.北京:清华大学出版社,2011.
[27] 胡云.RSA 算法研究与实现[硕士学位论文].北京:北京邮电大学,2010.
[28] 荣辉桂,莫进侠,常炳国,孙光,龙飞.基于 Shamir 秘密共享的密钥分发与恢复算法.通信学报,2015,36(3):60−69.

侯瑞涛(1993−),男,博士生,主要研究领域为数字水印,云存储安全.

李京(1971−),女,讲师,主要研究领域为云计算,电子商务安全.

咸鹤群(1979−),男,博士,副教授,CCF 高级会员,主要研究领域为云存储安全,区块链,隐私保护,密码学.

狄冠东(1994−),男,硕士生,主要研究领域为数字水印,云存储安全.