一種應用於挖掘關聯式法則可防止洩密的資料擾亂方法

賈坤芳
國立中興大學資訊科學研究所
kfjea@cs.nchu.edu.tw

賴俐錦
國立中興大學資訊科學研究所
s9056032@cs.nchu.edu.tw

摘要

資料挖掘主要的目的是從大量資料中挖掘出有用的資訊，但這樣的技術可能會危害到資料庫的安全，導致資料庫中的敏感資料(敏感項目集或敏感規則)被挖掘出來。針對敏感項目集的問題，目前已提出的方法都是直接對包含敏感項目集之交易來進行修改，這些方法雖可解決敏感項目集洩露的問題，但卻會造成部分非敏感項目集無法被挖掘出來。為了解決上述問題，本文提出一種資料擾亂的方法，此方法是利用“交易分解”的方式來拆解交易中的敏感項目集，達到隱藏的目的。實驗結果證實，利用這些分解後的交易來進行挖掘時，既不會造成敏感項目集被挖掘出來，亦不會造成非敏感項目集無法被挖掘出來。

關鍵字：資料庫安全、資料挖掘、敏感項目集、資料擾亂、交易分解

Keyword：database security、data mining、sensitive itemset、data perturbation、transaction decomposition

一、前言

資料庫中儲存各種重要的資料，若遭到有心人士竊取或竄改，則可能會危害到資料庫的安全，導致其他使用者的權益受損，因此有資料庫安全(database security)研究領域之興起。所謂資料庫安全是指保護資料庫中資料的安全性，以防止資料遭受未經授權的洩露(disclosure)、修改(alternation)及破壞(destruction)。然而資料挖掘(data mining)技術的發明，卻對資料庫安全帶來很大的衝擊[4]。

針對關聯式挖掘法(association rules mining)對資料庫所造成的安全危害，主要有二個議題。第一個議題[6][8][10]在探討如何挖掘分散於各個 site 的資料，又不會洩露各個 site 所擁有的資料，亦即隱私性(privacy)的問題。第二個議題[3][4][5][7][9][11]在探討藉由挖掘技術，導致資料庫中敏感資料被洩露出來，對資料庫的秘密性(secrecy)造成衝擊。

在第二個議題中，根據敏感資料的差異而有不同的解決方法。其中[3][9]探討如何隱藏敏感、重要的項目集 (sensitive itemset)的問題；[4][5][7][11]更進一步探討如何隱藏敏感規則(sensitive rule)的問題。雖然這些方法都可解決敏感資料洩露的問題，但卻也延伸出另外二個問題：1) 造成原本不存在(錯誤)的資料被挖掘出來，稱為 artificial pattern 的問題，2) 造成原本應挖掘出來的資料卻挖掘不出來，稱為 miss cost 的問題。

本文針對防止敏感項目集被挖掘出來的問題，提出一種新的資料擾亂(data perturbation)的方法，透過“交易分解(transaction decomposition)”的方式來分解具有敏感項目集的交易，使非敏感項目集的計數值正確無誤，敏感項目集的計數值降為 0。因此，本方法既可解決資料挖掘時敏感項目集洩露的問題，又不會產生 miss cost 及 artificial pattern 等問題。

本文之結構如下：第一節為前言，第二節探討防止藉由關聯式挖掘技術去挖掘出敏感資料之相關研究，第三節提出一種新的資料

擾亂方法以隱藏敏感項目集，第四節為實驗結果與分析。最後，第五節為結論。

二、相關研究

目前隱藏敏感資料的方法中，[3][9]探討防止敏感項目集洩露的問題，[4][5][7][11]探討防止敏感規則洩露的問題。

[3][9]提出利用降低敏感項目集的支持度來達到隱藏敏感項目集的目的。若交易中包含某個敏感項目集，則表示此筆交易中會出現這個敏感項目集，使得這個敏感項目集的計數值多 1。因此[3][9]提出透過刪除交易中敏感項目集內的某個屬性，讓交易不再出現此敏感項目集，避免被使用者挖掘出來。雖然[3][9]可解決敏感項目集洩露的問題，但卻會產生 miss cost 的問題，亦即部分非敏感資料也無法從資料庫中被挖掘出來。

延伸[3][9]的問題，[4][5][7][11]更進一步探討如何隱藏敏感規則。其中[5]根據關聯式規則的定義提出三種隱藏敏感規則的方法。前二種作法是從信賴度著手，分別藉由增加規則的前項支持度及減少規則的後項支持度來降低敏感規則的信賴度，而第三種作法是從支持度著手，藉由降低規則支持度使其低於設定之最小支持度。雖然[5]可解決敏感規則洩露的問題，但卻會產生 miss cost 及 artificial pattern (不存在的資料從資料庫中被挖掘出來)等問題。

針對 artificial pattern 的問題，[11]認為在某些情況下，給予使用者一些錯誤或不存在的規則可能會造成無法彌補的過錯；以醫療資料庫為例，若醫療人員參考到不正確的資訊，可能會危害到病人的生命。因此[11]提出在隱藏規則時，利用未知值(unknown value)來取代錯誤的值(false value)，解決 artificial pattern 的問題。但此法為了配合新加入的未知值，導致使用者之挖掘演算法必須修改，造成使用不便。

三、資料擾亂方法

本文主旨在探討關聯式挖掘法所造成敏感項目集洩露的問題。首先在 3.1 節敘述我們的問題，3.2 節提出新的資料擾亂的方法，在 3.3 節描述依此方法所設計的演算法 STD。最後在 3.4 節分析 STD 演算法的安全性。

3.1 問題描述與術語定義

敘述問題之前，我們先對一些相關名詞加以定義 [3][9]：

定義 1：**交易資料庫(transaction database，TDB)**是由多筆交易所組成，而每筆交易是由數個屬性集合而成。

在本文中，我們利用一個 Trans_DB 表格來儲存交易，此表格包含交易及計數值二個欄位，其中第一個欄位代表交易的內容，第二個欄位代表此交易出現的次數。

定義 2：**敏感資料集(sensitive data set，SDS)**是由多筆敏感項目集(sensitive itemset)所形成的集合，每筆敏感項目集代表一項不允許被挖掘出來的敏感資料。而可允許被挖掘出來的項目集我們稱為**非敏感項目集(non-sensitive itemset)**。

定義 3：若一筆交易中只包含一個敏感項目集則稱為**敏感交易(sensitive transaction)**；若包含一個以上的敏感項目集則稱為**衝突交易(conflicting transaction)**。

定義 4：**淨除(sanitization)**是指移除交易資料庫內所有交易所包含之敏感項目集的過程。經淨除產生的資料庫我們稱為**已淨除資料庫(sanitized database，SDB)**。而已淨除資料庫中的每一筆交易我們稱為**已淨除交易(sanitized transaction)**。

由於透過淨除產生之已淨除資料庫來進行挖掘時，可能會產生 hiding failure、miss cost

及 artificial pattern 等問題，以下為這三個問題的定義[9]：

定義 5：**hiding failure** 代表敏感資料從已淨除資料庫中被挖掘出來。

定義 6：**miss cost** 代表非敏感資料無法從已淨除資料庫中被挖掘出來。

定義 7：**artificial pattern** 代表錯誤或不存在的資料從已淨除資料庫中被挖掘出來，即原先在交易資料庫中挖掘不出來的資料，卻可從已淨除資料庫中挖掘出來。

本研究所探討的問題定義如下：給予一個交易資料庫及一個敏感資料集，如何將這個交易資料庫轉換成已淨除資料庫，並保證挖掘已淨除資料庫時，不會造成 hiding failure、miss cost 及 artificial pattern 等問題？由於知道每個項目集的計數值就可進行挖掘，因此本研究假設挖掘演算法與資料庫系統之介面為只能透過 SQL 的 aggregate function 來讀取已淨除資料庫中的資料計數值，如此可提高已淨除資料庫的安全。

3.2 資料擾亂方法

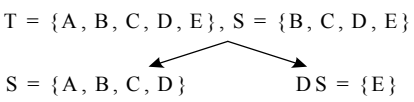
本節首先敘述如何解決敏感交易（交易只包含一個敏感項目集）所造成的洩露問題，然後敘述如何處理衝突交易（交易包含一個以上的敏感項目集）所造成的洩露問題。

3.2.1 敏感交易的淨除

3.2.1.1 “交易分解”方法

由於敏感交易會造成所包含的敏感項目集被洩露出來，因此我們須將交易中敏感項目集的部分切割出來進行拆解，以解決敏感項目集洩露的問題。而交易中非敏感項目集的部分可忽略不拆解，因為它們不會造成敏感項目集洩露的問題。

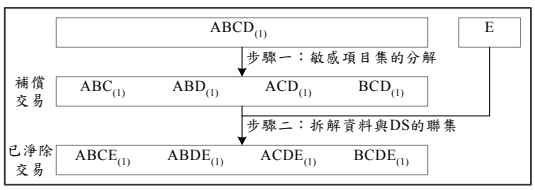
首先我們將一筆敏感交易 T 所包含的屬性拆成 *S* 及 *DS* 二個集合，分別代表該交易所包含的敏感項目集與非敏感項目集，*DS*＝*T*－*S*。以一筆長度為 5、計數值為 1 的敏感交易 ABCDE 為例，假設此筆敏感交易包含一個敏感項目集 ABCD，則此交易可拆解出下列二個集合(如圖一所示)。



圖一 敏感交易的切割

我們提出的“交易分解”方法共有二個步驟，首先在步驟一中處理切割出來的敏感項目集 *S*，將長度為 *|S|* 的敏感項目集分解出 *C(|S|,|S|-1)* 筆長度為 *|S|-1* 的項目集，在此我們用補償交易(compensation transaction，*CT*)來代表這些被分解出的項目集，補償交易之計數值等於原先敏感交易之計數值。在步驟二中將步驟一分解出的補償交易與 *DS* 進行聯集，產生已淨除交易來取代原先資料庫中的這筆敏感交易，而這些已淨除交易之計數值等於原先補償交易之計數值。利用此種方式可確保產生的已淨除交易都不包含任何敏感項目集，達到隱藏敏感項目集的目的。

延續圖一之例，敏感項目集 ABCD 透過分解會產生 *C(4,3)* 筆長度為 3 的補償交易 ABC、ABD、ACD 及 BCD，其中補償交易右下角括號內的數字代表其計數值。這些補償交易聯集 *DS*＝{E} 會產生已淨除交易 ABCE、ABDE、ACDE 及 BCDE，其中已淨除交易右下角括號內的數字代表它的計數值。在此例中，敏感項目集 ABCD 及其 superset ABCDE 皆不包含於任一個分解產生的已淨除交易，故可達到隱藏敏感項目集 ABCD 的目的。



圖二 敏感交易 ABCDE 透過“交易分解”來產生已淨除交易

雖然上述“交易分解”方法可達到隱藏敏感項目集的目的，但卻造成部分非敏感項目集之計數值有誤。其原因是我們將敏感交易分解後的項目集視為是新的交易(已淨除交易)，取代原先資料庫中的這筆敏感交易，然而這些已淨除交易經各個敏感項目集的輾轉分解，會造成敏感交易之子項目集的“來源增加”，導致計數值多計而產生 artificial pattern 的問題。

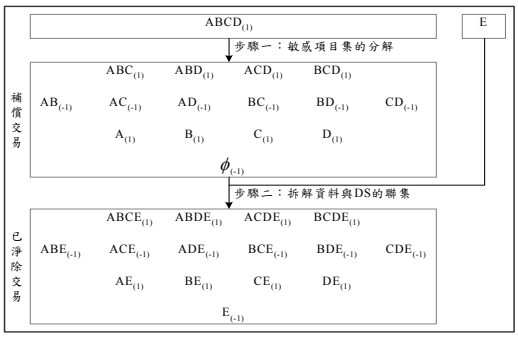
為了要確保交易分解後，所有非敏感項目集之計數值正確無誤，我們在進行“交易分解”時，須將造成計數值多計的問題考慮進來，下節探討如何解決此一問題。

3.2.1.2 修正的“交易分解”方法

為確保非敏感項目集之計數值正確無誤，我們提出一種“一加一減”的“交易分解”方法，其作法如下：首先在步驟一中處理切割出來的敏感項目集 S，將長度為$|S|$的敏感項目集之所有真子集都分解出來，而將這些長度為 $|S|$-k 的補償交易之計數值設定為：當 k 是奇數時將補償交易視為**加項**，其計數值等於原先敏感交易之計數值×(1)；當 k 是偶數時將補償交易視為**減項**，其計數值等於原先敏感交易之計數值×(-1)。

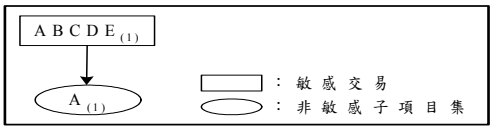
接下來在步驟二中將步驟一分解出的補償交易與 DS 進行聯集，產生已淨除交易來取代原先資料庫中的這筆敏感交易，而這些已淨除交易之計數值等於原先補償交易之計數值。根據這種“一加一減”的“交易分解”方法，一筆敏感交易經逐層分解後，可分解出 $\left(\sum_{k=1}^{|S|} C(|S|,|S|-k)\right)$ 筆已淨除交易來取代它，即分解出 $\left(2^{|S|}-1\right)$ 筆已淨除交易。

延續圖一之例，敏感交易 ABCDE 透過“一加一減”的“交易分解”方法，會分解出 $C(4,3)+C(4,2)+C(4,1)+C(4,0)=2^4-1=15$ 筆已淨除交易，如圖三所示。在此例中，敏感項目集 ABCD 及其 superset ABCDE 皆不包含於任一個分解產生的已淨除交易，達到隱藏敏感項目集 ABCD 的目的。

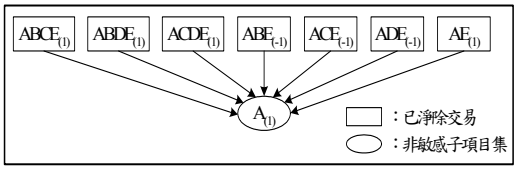


圖三 敏感交易 ABCDE 透過“一加一減”的“交易分解”來產生已淨除交易

透過上述“一加一減”的“交易分解”方法，可確保所有非敏感子項目集之計數值正確無誤。以長度為 1 的非敏感子項目集 A 為例，圖四(a)表示原先敏感交易 ABCDE 使得子項目集 A 之計數值多 1，圖四(b)表示逐層分解產生的已淨除交易使得子項目集 A 之計數值多 1，故可知執行交易分解前與交易分解後不會造成子項目集 A 之計數值錯誤。



圖四(a) 敏感交易使非敏感子項目集 A 之計數值多 1



圖四(b) 已淨除交易使非敏感子項目集 A 之計數值多 1

3.2.2 衝突交易的淨除

在包含一個以上的敏感項目集的衝突交易處理上，我們重複使用上述“一加一減”的“交易分解”方法，一次對一筆衝突交易中的一

個敏感項目集進行淨除。如果一筆衝突交易包含 n 個敏感項目集，則執行 n 個 pass 的“交易分解”，依序將每一個被包含的敏感項目集拆解掉(即一個 pass 只處理一個敏感項目集)。

衝突交易的處理步驟如下，首先將一筆衝突交易視為是一個敏感交易，執行“一加一減”的“交易分解”，來將其中第一個敏感項目集分解掉，以產生半淨除交易。接下來將這些半淨除交易視為敏感交易，繼續將其中第二個敏感項目集分解掉，重複執行上述動作，直到所有的敏感項目集都已從半淨除交易中分解完畢為止。最後，將產生的已淨除交易輸出，取代原先資料庫中的這筆衝突交易。由於衝突交易中包含多個敏感項目集，因此每次執行“交易分解”所產生的結果可能含有其他未處理的敏感項目集，在此我們用半淨除交易(semi-sanitized transaction，*SST*)來代表中間分解的結果。詳細作法請參考[1]。

處理衝突交易內之敏感項目集時，我們依據敏感項目集長度由短到長的次序來進行分解(即先處理短的敏感項目集，再處理長的敏感項目集)。因為敏感交易所包含的敏感項目集長度愈長，則執行“交易分解”所產生的半淨除交易愈多(分解出 $2^{|S|}-1$ 筆)。若敏感項目集的分解次序是採用由長到短的順序，則一開始會產生較多的半淨除交易，然後這些半淨除交易都須再與後面較短的敏感項目集進行比對，故須分解的次數愈多。因此在執行淨除之前，我們先將敏感資料集 *SDS* 根據敏感項目集的長度由短到長來排序，降低分解的成本。

3.3 STD 演算法

交易分解的演算法 STD，如圖五所示，其中符號 Q 代表緩衝區，用來暫存敏感交易或衝 Q 中正在處理的交易；S 代表目前正在處理的敏感項目集，其長度為$|S|$；*CT* 代表“交易分解”產生之補償交易，其長度為$|CT|$；*SST* 代表執行“交易分解”產生的半淨除交易；*count* 代表交易的計數值。函數 **transaction_sanitization()**執行 3.2.1.2 節的交易分解方法，詳細步驟參考[1]。

Algorithm STD

Input: 交易資料庫(*TDB*)、敏感資料集(*SDS*)

Output: 已淨除資料庫(*SDB*)

Method:

1. Sort sensitive data set *SDS* in ascending order of $|S|$;
2. **foreach** transaction $T \in TDB$ **do begin**
3. $Q = \phi$;
4. Insert T into Q ;
5. S = first sensitive itemset in *SDS* ;
6. **while**($S \neq NULL$) **do begin**
7. P = first transaction in Q ;
8. **while**($P \neq NULL$) **do begin**
9. **if** ($P \supseteq S$) **then call** transaction_sanitization(P, S) ;
10. P = next transaction in Q ;
11. **end while**
12. S = next sensitive itemset in *SDS* ;
13. **end while**
14. Output all transactions from Q to *SDB* ;
15. **end foreach**

STD 演算法的時間複雜度分析如下：若一筆原始交易包含一敏感資料，則會分解出 $\left(2^{|S|}-1\right)$個半淨除交易來取代緩衝區 Q 中的這筆交易，然後再將這些半淨除交易視為是敏感交易，繼續與後面的敏感項目集比對，直到所有敏感項目集都處理完畢為止，故執行每一筆交易的複雜度為 $O\left(\left(2^{|S|}-1\right)^{l}\right)$，其中$|S|$代表最長的敏感項目集長度，$l$ 代表交易所包含的敏感項目集平均個數。由於每筆敏感交易(或衝突交易)都會呼叫 transaction_sanitization()函數來

進行“交易分解”，因此整個 STD 演算法執行的複雜度為 $O\left(\mathrm{m}\times\left(2^{|S|}-1\right)^{\gamma}\right)$，其中 m 代表敏感交易(或衝突交易)的總筆數。

3.4 STD 的安全性分析

本節我們以 hiding failure、miss cost 及 artificial pattern 三個問題作為方法正確性及安全性的評估依據。“交易分解”方法 STD 具有以下二個性質[1]：

Lemma 1：“交易分解”方法不會造成非敏感項目集的計數值有誤。

Lemma 2：“交易分解”方法不會造成原先交易中的非敏感項目集遺失，亦不會產生不存在於交易中的項目集。

產生 hiding failure 問題的原因是：敏感項目集包含於已淨除交易中，使得敏感項目集之計數值大於挖掘時所設定之最小支持度，而導致敏感項目集被挖掘出來。由於 STD 已將敏感交易或衝突交易中所包含之敏感項目集都拆解掉，因此經淨除產生之已淨除交易中都不包含有任何敏感項目集，使得敏感項目集的計數值降為 0，因此不管挖掘門檻值(最小支持度)如何調整，都不會造成敏感項目集被洩露出來。由此可知，STD 不會產生 hiding failure 的問題(證明請參考[1])。

產生 miss cost 問題的原因有二：一是淨除方法造成原先交易中的非敏感項目集遺失；二是淨除方法造成原先交易中的非敏感項目集之計數值少計。根據 Lemma 1 及 Lemma 2 可知，STD 不會有上述二個問題，故不會產生 miss cost 的問題(證明請參考[1])。

產生 artificial pattern 問題的原因有二：一是淨除方法造成原先不存在的非敏感子項目集被產生出來；二是淨除方法造成原先交易中的非敏感項目集之計數值多計。根據 Lemma 1 及 Lemma 2 可知，STD 不會有上述二個問題，故不會產生 artificial pattern 的問題(證明請參考[1])。

四、實驗結果與分析

本節說明執行“交易淨除”方法的實驗與分析。我們使用 Pemtium 1.4 GHZ CPU、128MB RAM 及 40GB Disk 的實驗平台，作業系統為 Red hat linux 7.1，並利用 C language 作為程式撰寫的工具。

我們利用 IBM 的資料庫產生程式來產生實驗用的交易資料庫，共產生五個交易資料庫分別為 T5I4D10KN1K、T5I4D20KN1K、T5I4D30KN1K、T5I4D40KN1K 及 T5I4D50KN1K，其中參數 T 代表資料庫中的交易平均長度、I 代表資料庫中的候選項目集平均長度、D 代表資料庫中的交易總筆數，N 代表資料庫中的屬性個數。

和[9]一樣，我們以亂數來產生實驗用的敏感資料集，首先利用 Apriori[2]演算法對上述產生的交易資料庫進行挖掘，挖掘時將最小支持度設定成 0.5%，然後利用亂數選取的方式，從高頻項目集中取出 *SD* 筆長度為 *SL* 的資料，並將這些取出的高頻項目集視為是後續所欲隱藏的敏感項目集，參數 *SD* 代表敏感項目集總筆數，參數 *SL* 代表敏感項目集長度。

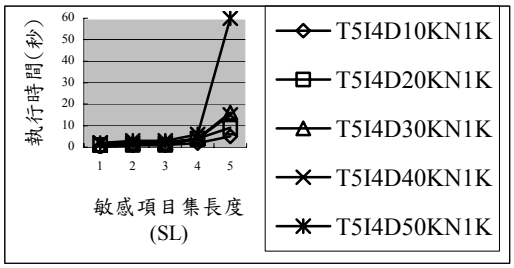
為了驗證本方法的正確性及了解執行效能，我們共設計了三組實驗。第一組實驗驗證淨除方法的正確性，其結果列於表一。表一顯示原始交易資料庫 T5I4D10KN1K 及已淨除資料庫 T5I4D10KN1K-SD100SL5，在最小支持度為 0.5%時，利用 Apriori[2]演算法所挖掘出的高頻項目集個數，我們用 *Lk* 代表挖掘出長度為 *k* 的高頻項目集之個數。其中已淨除資料庫 T5I4D10KN1K-SD100SL5 是原始交易資料庫 T5I4D10KN1K 在敏感項目集筆數為 100、敏感項目集長度為 5 時，執行 STD 所產生的。

表一 交易資料庫 T5I4D10KN1K 及已淨除資料庫 T5I4D10KN1K-SD100SL5 執行挖掘結果

	T5I4D10KN1K	T5I4D10KN1K -SD100SL5
L1	884	884
L2	6466	6466
L3	3977	3977
L4	3113	3113
L5	2024	1924
L6	1100	816
L7	478	158
L8	152	6
L9	31	
L10	3	

在 *L1*、*L2*、*L3* 及 *L4* 的情況下，T5I4D10KN1K 及 T5I4D10KN1K-SD100SL5 二者挖掘出的高頻項目集個數相同；在 *L5* 的情況下，T5I4D10KN1K-SD100SL5 挖掘出的高頻項目集個數比 T5I4D10KN1K 挖掘出的個數少 100 個；在 *L6*、*L7*、*L8*、*L9* 及 *L10* 的情況下，T5I4D10KN1K-SD100SL5 挖掘出的高頻項目集個數比 T5I4D10KN1K 挖掘出的個數還少。經比對發現除了敏感項目集及其 superset 之外，二者所挖掘出的高頻項目集內容完全相同，故可知本方法不會造成敏感的高頻項目集被挖掘出來，亦不會造成非敏感的高頻項目集挖掘不出來，故本方法不會產生 hiding failure、miss cost 及 artificial pattern 等問題。

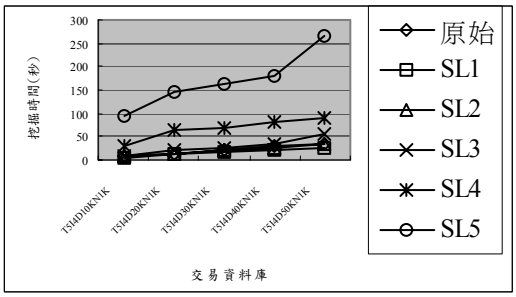
第二組實驗進行產生已淨除資料庫的效能測試，其結果列於圖六中。在實驗的過程中，將敏感項目集筆數設定為 100，敏感項目集長度分別為 1、2、3、4、5 時，測量資料庫 T5I4D10KN1K、T5I4D20KN1K、T5I4D30KN1K、T5I4D40KN1K 及 T5I4D50KN1K 執行 STD 演算法所需的時間。



圖六 各種資料庫大小及敏感項目集長度下執行 STD 的時間數據圖

由圖六可發現，執行 STD 所花費的時間隨著敏感項目集長度增加而增加，其原因是一筆敏感交易或衝突交易中所包含的敏感項目集長度愈長，則演算法 STD 執行"交易分解"的次數愈多(複雜度為 $O\left(2^{SL}-1\right)^{l}$，其中 l 代表交易所包含的敏感項目集平均個數)，故 STD 執行的時間愈長。由此實驗可知，演算法 STD 的執行時間與敏感項目集長度成指數成長，然而因演算法 STD 只執行一次，一旦淨除完成，使用者即可對此已淨除資料庫進行挖掘。

第三組實驗測試挖掘已淨除資料庫的效能，其結果列於圖七。實驗中，敏感項目集筆數設定為 100，敏感項目集長度分別為 1、2、3、4、5 時，測量由 T5I4D10KN1K、T5I4D20KN1K、T5I4D30KN1K、T5I4D40KN1K 及 T5I4D50KN1K 產生的已淨除資料庫所需的挖掘時間。



圖七 原始資料庫與不同長度的敏感項目集所產生之已淨除資料庫，執行挖掘的時間數據圖

由圖七可發現，當敏感項目集長度(*SL*)大於 2 時，所產生之已淨除資料庫執行挖掘的時間比針對原始交易資料庫執行挖掘的時間還高，且 *SL5* 與 *SL4* 的執行時間差距大於 *SL4* 與 *SL3* 的執行時間差距。其原因是我們是採用

“交易分解”的作法來解決敏感項目集洩露的問題，當敏感交易或衝突交易中所包含的敏感項目集長度愈長，則演算法 STD 執行“交易分解”的次數愈多(複雜度為 $O\left(2^{SL}-1\right)^{l}$，其中 l 代表交易所包含的敏感項目集平均個數)，因此分解出的已淨除交易愈多，導致已淨除資料庫中交易筆數成指數增加。又已淨除資料庫中的交易筆數愈多，則執行資料挖掘所需掃描資料庫的成本愈高，因此執行挖掘的時間愈長。

由上述三組實驗可知，本方法花費許多時間來進行資料庫的修改，且會導致挖掘演算法的執行效能降低，但本方法具有高度的安全性(沒有 hiding failure 的問題[1])及高度的準確性(沒有 miss cost 及 artificial pattern 等問題)。

五、結論

在隱藏敏感項目集的問題上，傳統的方法皆無法完全解決此一問題。它們可能解決了 hiding failure 的問題，卻引發 miss cost 的問題。本研究從另一個角度切入，提出了一種“交易分解”的方法，將每一筆敏感交易所包含的敏感項目集拆解掉，使得敏感項目集的計數值降為 0、非敏感項目集的計數值不變，同時解決 hiding failure、miss cost 及 artificial pattern 這三個問題。

目前本文提出之“交易分解”方法只能解決敏感項目集的問題，無法直接適用於隱藏敏感規則。未來將修正“交易分解”的方法，以解決敏感規則洩露的問題。

六、參考文獻

[1] 賴俐錦, 一種應用於挖掘關聯式法則可防止洩密的資料擾亂方法, 碩士論文, 國立中興大學, 2003.

[2] R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” Proceedings of the IEEE International Conference on Data Engineering Workshop, 1998, pp. 402-411.

[3] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim and V. Verykios, “Disclosure Limitation of Sensitive Rules,” Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop, 1999, pp. 45-52.

[4] C. Clifton, “Using Sample Size to Limit Exposure to Data Mining,” Journal of Computer Security, vol. 8, no. 4, 2000, pp. 281-307.

[5] E. Dasseni, V. Verykios, A. Elmagarmid and E. Bertino, “Hiding Association Rules by Using Confidence and Support,” Proceedings of the 4th International Information Hiding Workshop, 2001, pp. 369-383.

[6] A. Evfimievski, R. Srikant, R. Agrawal and J. Gehrke, “Privacy Preserving Mining of Association Rules,” Proceedings of the 8th ACM SIGKDD International Conference, 2002, pp. 217-228.

[7] T. Johnsten, V. Raghavan and K. Hill, “The Security Assessment of Association Mining Algorithm,” Proceedings of the 16th Annual IFIP WG 11.3 Working Conference, 2002, pp. 1-14.

[8] M. Kantarcioglu and C. Clifton, “Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data,” Proceedings of the ACM SIGMOD Workshop, 2002, pp. 24-31.

[9] S. Oliveira and O. Zaïane, “Privacy Preserving Frequent Itemset Mining,” Proceedings of the IEEE International Conference on Data Engineering Workshop, 2002, pp. 43-54.

[10] S. Rizvi and J. Haritsa, “Privacy-Preserving Association Rule Mining,” Proceedings of the 28th International Conference on Very Large Database, 2002, pp. 20-23.

[11] Y. Saygin, V. Verykios and C. Clifton, “Using Unknowns to Prevent Discovery of Association Rules,” ACM SIGMOD Record, vol. 30, no. 4, 2001, pp. 45-54.