T-79.515 Special Course on Cryptology: Privacy-Preserving Frequent Itemset Mining on Horizontally Distributed Data

Sven Laur

February 16, 2004

Abstract

This survey covers some security aspects of cooperative frequent itemset mining. We observe scenarios, where individual records of the database are distributed among different parties. All parties are willing to cooperate in order to find globally frequent itemsets, but they do not want to reveal too much information. The survey examines one possible solution proposed in article [KC02], but an alternative based on Benaloh's protocol [Ben87] is briefly discussed.

1 Introduction: motivation

A classical application of frequent itemset mining is an analyse of supermarket data. Discovered frequent itemsets characterize behavior of customers—frequent sets denote objects often bought together. Moreover, refined analyse can reveal causal relatioships between various items. For example customers, who buy pizza and beer, tend to buy chips. Discovery of new novel causal relations makes frequent itemset mining apealing in many applications.

In order to formalize the underlying idea, we introduce simple notations. Imagine that we have a database $\mathcal{DB}$ and each record R is a set of items. All possible items $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m$ form the itemset I and frequent itemsets $\mathcal{A} = \{\mathcal{I}_{i_1}, \ldots, \mathcal{I}_{i_k}\}$ are just subsets of I. The support of an itemset $\mathcal{A}$ is the number of database records that contain those items, formally

$$\mathrm{supp}(\mathcal{A}) = \#\left\{R \in \mathcal{DB} \;:\; \mathcal{A} \subseteq R\right\}.$$

We call an itemset frequent, if the support is over some fixed threshold κ.

Since frequent itemsets descirbe only items bought together, we need a refinment that indicates causal relations like $Coke \Rightarrow Chips$. The usefulness of such rules are described by two parameters: support and confidence. The support of an assotciation rule $\mathcal{A} \Rightarrow \mathcal{B}$ characterizes how often the rule is supported by data and the confidence quantifies the uncertainty $p(\mathcal{B}|\mathcal{A})$. Formal definitions are

$$\mathrm{supp}(\mathcal{A} \Rightarrow \mathcal{B}) = \mathrm{supp}(\mathcal{A} \cup \mathcal{B}),$$
$$\mathrm{conf}(\mathcal{A} \Rightarrow \mathcal{B}) = \frac{\mathrm{supp}(\mathcal{A} \cup \mathcal{B})}{\mathrm{supp}(\mathcal{A})}.$$

The rules with low support occur rarely and are usually caused by idiocracies of the data, thus uninteresting. It is easy to see that rules with high enough support can be computed from listing of frequent itemsets. Thus mining frequent itemsets is sufficient for exploring association rules.

Usually large collections of data are owned by different departments or even by different corporations. There are two possibilities: a horiontal or a vertical division. In case of horizontal division the individual records are not divided, wheras the vertical division means that each record is splited. We consider only horizontal division among t parties

$$\mathcal{DB} = \mathcal{DB}_1 \cup \mathcal{DB}_2 \cup \ldots \cup \mathcal{DB}_t.$$

Clearly, each party can compute local summaries, but is unable to explore global trends. If the data belongs to rival parties (competing companies), then parties are unwilling to share the data. Even more, laws might prohibit to reveal any delicate information to third parties. Therefore, all parties

want strict security guarantees that their data and local meta-data remains private. Of course, the summary itself leaks some private information, but this is unavoidable. Recent results [Mie03] indicate that restoring database based on frequent itemset data is generally intractable.

Another interesing scenario is a covernment data acquisition, where one well-established party is doing datamining and will later publish summaries. Here, we can trust that published summaries are correct ones, but must still consider an information leakage caused by data mining protsess.

2 Briefly about Apriori

Most of the frequent itemset algorithms are slight modifications of the Apriori [AIS93, AMS+96] algorithm. The Apriori uses explicitly anti-monotone relation between itemsets and their supports, formally described by the rule

$$\mathcal{A} \subseteq \mathcal{B} \quad \Longrightarrow \quad \mathrm{supp}(\mathcal{A}) \geq \mathrm{supp}(\mathcal{B})$$

The rule follows immediately, since all records containing items $\mathcal{B}$ contain $\mathcal{A}$ as well. Shortly put, all subsets of the frequent itemset are also frequent. So instead of searching over all possible subsets, the algoritms starts from one-element itemsets. Then tests only those two-element sets that have frequent subsets and so on. It is easy to see that for candidate $\mathcal{B}$ with cardinality ℓ it is sufficient to check whether all $\ell - 1$ element subsets are frequent. Algorithm 1 is ascetic description of the Apriori.

Algorithm 1: Apriori algorithm

Input: Support threshold κ.
♠♠♠ No frequent sets, candidate set I ♠♠♠
$F = \emptyset$; $C = I$; $\ell = 1$
while $|C| > 0$ **do**
  ♠♠♠ Find all valid candidates ♠♠♠
  $F = F \cup \{\mathcal{A} \in C : \ \mathrm{supp}(\mathcal{A}) > \kappa\}$
  ♠♠♠ Form ℓ-element candidate set ♠♠♠
  $C = \{\mathcal{B} \in \mathcal{P}(I) : \ |\mathcal{B}| = \ell, \ \mathcal{A} \subseteq \mathcal{B} \Rightarrow \mathcal{A} \in F\}$
  $\ell = \ell + 1$
return F

The size of output is determined by the threshold κ. If κ is small then the number of frequent itemsets is large. The working time depends linearly on the size of database $|\mathcal{DB}|$ and on the output size $|F|$. In that sense the Apriori algorithm is optimal. Of course, the Apriori will slow down, if there is even a single large frequent set. Therefore, the Apriori is suitable for mining sparse records with no large frequent itemsets.

3 Secure distributed mining

Now consider a horizontally distributed database shared by t parties $\mathcal{DB} = \mathcal{DB}_1 \cup \cdots \cup \mathcal{DB}_t$. Instead of global supports parties can compute local supports $\mathrm{supp}_i(\mathcal{A})$. If the number of all records $n = |\mathcal{DB}|$ and local records $n_i = |\mathcal{DB}_i|$, then the following implication holds

$$\mathrm{supp}(\mathcal{A}) > \kappa \quad \Rightarrow \quad \exists i : \ \mathrm{supp}_i(\mathcal{A}) > \frac{n_i \kappa}{n} = \kappa_i.$$

Moreover, we have three classes of frequent itemsets: globally and locally frequent itemsets, and locally supported globally frequent itemsets

$$\begin{aligned} F &= \{\mathcal{A} : \ \mathrm{supp}(\mathcal{A}) \ > \kappa\}, \\ F_i &= \{\mathcal{A} : \ \mathrm{supp}_i(\mathcal{A}) > \kappa_i\}, \\ LF_i &= F \cap F_i. \end{aligned}$$

The distributed frequent itemset mining is based on dependecies between F and LF_i. If $\mathcal{B}$ be a globally frequent itemset, then there is a site i such that $\mathrm{supp}_i(\mathcal{B}) > \kappa_i$. Hence, all subsets of $\mathcal{B}$ are locally and globally frequent and we have the implication

$$\mathcal{B} \in F \quad \Longrightarrow \quad \exists i : \ \mathcal{A} \subseteq \mathcal{B} \Rightarrow A \in LF_i.$$

In other words, we can create the Apriori candidate list locally. Each party generates candidate sets C_i based on locally supported global sets LF_i, then tests them locally and afterwards a general candidate list C is formed as a union of new elements in LF_i. Algorithm 2 formalizes the idea. Our next aim is to modify the algorithm so that minimal amount of information is leaked.

There are two different approaches to obtain security. Secure multi-party computation (MPC) requires that only the end-result F should become public. The MPC approach tends to force inefficient protocols. Therefore, Clifton and Kantarcioglu relaxed the setting allowing some information become public. Namely, the global candidate set C and some additional details will leak out.

Algorithm 2: Distributed Apriori algorithm

Input: Normalized support threshold κ/n.

♠♠♠ Calculate local threshold ♠♠♠
$\kappa_i = n_i\kappa/n$
♠♠♠ No frequent sets, candidate set I ♠♠♠
$F = \emptyset$; $LF_i = \emptyset$; $C_i = I$; $C = I$; $\ell = 1$
while $|C| > 0$ **do**
  ♠♠♠ Find all valid candidates ♠♠♠
  $F_i^* = \{\mathcal{A} \in C_i : \ \mathrm{supp}_i(\mathcal{A}) > \kappa_i\}$
  ♠♠♠ Broadcast candidates ♠♠♠
  $C = F_1^* \cup \cdots \cup F_t^*$
  ♠♠♠ Global test ♠♠♠
  $F = F \cup \{\mathcal{B} \in C : \ \mathrm{supp}(\mathcal{B}) > \kappa\}$
  $LF_i = LF_i \cup \{\mathcal{B} \in C : \ \mathcal{B} \in F, \ \mathcal{B} \in F_i^*\}$
  ♠♠♠ New local candidate set ♠♠♠
  $C_i = \{\mathcal{B} \in \mathcal{P}(I) : \ |\mathcal{B}| = \ell, \ \mathcal{A} \subseteq \mathcal{B} \Rightarrow \mathcal{A} \in LF_i\}$
  $\ell = \ell + 1$
return F

Whether the leak is important depends entirely on the application.

We have to modify the broadcast and global testing phase of Algorithm 2, since other steps do not depend on local data. In other words, we must compute securely unions and sums and test inequalities. Finding unions and sums is rather easy, since several well-known cryptographic solutions exist.

Yet another question is whether, we allow malicious parties who deviate from a specified protocol or not. Clearly, it is easier to device solutions for semi-honest model, where everybody follows the protocol, than garantee security against malicious behavior. In the following, we examine only a semi-honest behavior.

4 Private union protocols

Clifton and Kantarcioglu proposed a naive union protocol that is secure in the semi-honest model, if there is no collusion between different parties. The protocol explicitly assumes that there exists a public superset X that contains all possible inputsets C_i of dissipating parties. Also, the upper bound d of the cardinality C_i is public. In our case the global Apriori candidate set is suitable for X.

Since X is public we can encode elements $\mathcal{A}$ of C_i by binary words. In order to hide the cadinality of C_i, we need a padding set F consisting of unused binary words. In the following Algorithm 3, we are explicitly dealing with encodings.

Algorithm 3: CK private union protocol

Public input: Superset X and count d.
Private input: Set $C_i = \{\mathcal{A}_1, \cdots, \mathcal{A}_{k_i}\}$.
Public output: Union $C = C_1 \cup \cdots \cup C_t$.

♠♠♠ Pad C_i with elements of F ♠♠♠
Set $M = C_i$.
Add false elements to M so that $|M| = d$.
♠♠♠ Sequential encryption ♠♠♠
for $j = 1$ ***to*** $t - 1$ **do**
  Encrypt $\mathsf{E}_i(M) = \{\mathsf{E}_i(\mathcal{A}) : \mathcal{A} \in M\}$.
  Send $\mathsf{E}_i(M)$ to right neighbor.
  Receive result M from left neighbor.
if *i even number* **then**
  Send $\mathsf{E}_i(M)$ to party 2.
else
  Send $\mathsf{E}_i(M)$ to party 1.
♠♠♠ Duplicate elimination ♠♠♠
Parties 1 and 2 eliminate duplicates.
Party 2 sends all messages to 1.
Party 1 eliminates duplicates.
♠♠♠ Decryption phase ♠♠♠
Messages are decrypted $\mathsf{D}_t \ldots \mathsf{D}_1$.
Party t removes false elements and broadcast result.

The protocol is based on a commutative encryption scheme. Unfortunately, only few commutative schemes are known [**?**] and they are not fast. In Algorithm 3 each party has is own private key and can encrypt messages with the encryption function E_i and decrypth with D_i. If a message is encrypted iteratively, then the encryption order is not important or more formally

$$\mathsf{E}_1\mathsf{E}_2 \ldots \mathsf{E}_t(\mathcal{A}) = \mathsf{E}_{\pi(1)}\mathsf{E}_{\pi(2)} \ldots \mathsf{E}_{\pi(t)}(\mathcal{A}) \quad (1)$$

for all possible messages and permutations π. The probability of collisions (decryption failures)

$$\Pr\left[\mathsf{E}_1\mathsf{E}_2 \ldots \mathsf{E}_t(\mathcal{A}_1) = \mathsf{E}_{\pi(1)}\mathsf{E}_{\pi(2)} \ldots \mathsf{E}_{\pi(t)}(\mathcal{A}_2)\right],$$

when $\mathcal{A}_1 \neq \mathcal{A}_2$, should be negligible. The equation (1) allows parties 1 and 2 to eliminate duplicate encryptions of a same set element $\mathcal{A}$. However, it also allows to detect size of intersections between C_i and C_{i+2k}. The maximal information leakage is summarized in the following theorem.

Theorem 4.1 *The CK protocol privately computes the union if there are no colluding parties and reveals at most:*

- *size of all intersections* $|C_i \cap C_{i+2k}|$*;*
- *size of intersection* $|D_1 \cap D_2|$*;*
- *size of* $|D_1|$ *and* $|D_2|$*;*

where $D_1 = C_1 \cup C_3 \cup \ldots$ *and* $D_2 = C_2 \cup C_4 \cup \ldots$.

However, the privacy is guaranteed for a single execution, since re-execution allows parties 1 and 2 can distinguish repeating sets. The security quarantee is rather weak, since one can use protocols that are computationally more efficient and give a strict security guarantee.

First, if cardinalities of the superset X and the end-result C are approximately the same order, then voting for each element $\mathcal{A} \in X$ cause only a slight efficiency loss. The voting step itself can be implemented by secure multiplication.

Algorithm 4: Generic union protocol

Public input: Superset X.
Private input: Set $C_i = \{\mathcal{A}_1, \cdots, \mathcal{A}_{k_i}\}$.

$C = \emptyset$
for $\mathcal{A} \in X$ **do**
  if $\mathcal{A} \in C_i$ **then** $b_i = 0$ **else** $b_i = 1$
  Securely multiply $c \equiv b_1 \cdots b_t \mod 2$.
  if $c \neq 1$ **then**
    Add $\mathcal{A}$ to C.
return C

The correctness follows from De Morgan's law $b_1 \vee b_2 = \neg(\neg b_1 \wedge \neg b_2)$. The secure multiparty multiplication can be implemented with Benaloh' protocol. The protocol is fully described in the next section. It extreamly efficient since all calculations are done in $\mathbb{Z}_2$ and each party has to do $2t$ multiplications. Only drawback is relatively large communication complexity.

Theorem 4.2 *The generic union protocol that uses Benaloh' protocol for multiplication is unconditionally secure against coalition of* $t-1$ *parties.*

Since the answers to each voting are determined by C, the security of the generic union protocol is determined by voting substep. As the Benaloh' protocol is secure against coalition of $t-1$ parties, the claim follows.

5 Private addition and inequality test

For finding global supports $\text{supp}(\mathcal{A}) \geq \kappa$, we are faced with distributed inequality problem. Again Clifton and Kantarcioglu produce a solution assuming that there are no coalitions.

Algorithm 5: CK inequality test

Private input: Private support $\text{supp}_i(\mathcal{A})$.
Public input: Large modulus $m > 2n$ such that $\gcd(m, n) = 1$.

Party 1 chooses $r \in \mathbb{Z}_m$.
Sets $c \equiv \text{supp}_1(\mathcal{A}) + r - \kappa_1 \mod m$.
for $i = 1$ ***to*** t **do**
  $c \equiv c + \text{supp}_i(\mathcal{A}) - \kappa_i \mod m$.
Parties 1 and t use Yao's circuit and determine $?c - r \geq 0 \mod m$.

The condition $\gcd(m, n) = 1$ allows to embed fractional thresholds κ_i into $\mathbb{Z}_m$ so that

$$\kappa_1 + \kappa_2 + \cdots + \kappa_t \equiv \kappa \mod m.$$

Hence, $c - r \equiv \text{supp}(\mathcal{A}) - \kappa \mod m$. As latter is in the interval $(-n, n)$ the sign of $c - r$ determines whether $\text{supp}(\mathcal{A}) \geq \kappa$. Currently, only Yao's general circuit construction allows securely compute the predicate, but this is rather slow substep.

Coalitions of parties i and $i+2$ can determine the value of $\text{supp}_i(\mathcal{A})$ without revealing their shares, thus the non-collusion assumption is not realistic. However, if we relax the constraint and allow $\text{supp}(\mathcal{A})$ become public then we can use Benaloh' protocol for adding. First, consider a random matrix in additive group G

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1t} \\ a_{21} & a_{22} & \cdots & a_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ a_{t1} & a_{t2} & \cdots & a_{tt} \end{pmatrix}$$

where row sums a_i are fixed. Clearly, all proper subsets of row elements have uniform distribution. Therefore, column sums of $t-1$ arbitrary columns

have also uniform distribution. This forms the core of the Beneloh' protocol.

In the Benaloh' protocol, we first sum over the columns and then over the column sums and obtain the som of all matrix elements $a = a_1 + \cdots + a_t$.

Algorithm 6: Benaloh' protocol

Private input: Private term a_i.

Choose randomly $a_{i1} + a_{i2} + \cdots + a_{it} = a_i$.
for $j = 1$ ***to*** t **do**
  Send a_{ij} to the jth party.

Calculate column sum $b_i = a_{1i} + a_{2i} \cdots + a_{ti}$
Broadcast values b_i.
return $b_1 + b_2 + \cdots + b_t$.

Theorem 5.1 *The Beneloh' protocol is unconditionally secure against coalition up to $t-1$ parties.*

Proof. Let us renumber the matrix rows and columns so that honest parties have top-left rows and columns. To simulate view of the coalition, we first send random elements a_{ij} to coalition. Since the $t-1$ column sums have uniform distribution we can choose answers b_i of honest parties randomly except for the $b_1 = a - (b_2 + \cdots + b_t)$. Note that the simulator knows values $b_2, \ldots, b_t$ and the end-result a. Since the view is perfectly simulated the claim follows. ■

Clifton and Kantarcioglu state, that combining Benaloh's protocol parties 1 and 2 can securely obtain shares r and $\text{supp}(\mathcal{A}) - \kappa - r$ and then use Yao's circuit to evaluate predicate $\text{supp}(\mathcal{A}) \geq \kappa$. But collaborating parties 1 and 2 can reveal $\text{supp}(\mathcal{A})$. Hence, the security against collusion requires stronger protocol.

6 Impossibility of two-party protocol

Let us consider the two-party case. If the public outcome is a listing of frequent sets along with the supports, then even the ideal protocol leaks useful information. Namely, the the global suppport reveals local support of the other party. If the hostile party provides empty database or uniformly filled database, the he can deduce all frequent sets of the victim. In most cases, there is no feasible cryptographic mechanisms that could elliminate such threats, thus we cannot guarantee security at all.

If the listing of frequent itesets is without supports, then it is possible to achieve the goal by running the usual Apriori in conjunction with Yao's circuit for comparison. However, the approach will be computationally demanding and the same empty database trick will work. Hence, securing the two-party frequent itemset mining is questionable.

7 Open problems

Clearly, proposed solution [KC02] has limited applicability. Therfore, we need realistic attack scenarios that allow coalitions, denial of service attacks or even malicious behavior. Although the association rule mining can be reduced to frequent itemset mining, we need different kind of protocols if we do not vant to reveal global frequencies of itemsets. Also, a relatively efficient two or multiperty inequality test would be nice.

References

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328, 1996.

[Ben87] Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Advances in Cryptology - Crypro'86*, volume 263 of *LNCS*, pages 251–260, 1987.

[KC02] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining

of association rules on horizontally partitioned data, 2002.

[Mie03] Taneli Mielikäinen. On inverse frequent set mining, 2003.