# Private Itemset Support Counting

Sven Laur[†]
swen@math.ut.ee

Helsinki University of Technology

[†] Joint work with Helger Lipmaa and Taneli Mielikäinen

# What is frequent itemset mining?

Many datasets are inherently discrete:

- supermarket data
- various error log records—web-logs, network failures etc
- search indices used by Google, Spotlight etc
- medical and genetic databases

**Intuition:** Frequent patterns capture main characteristics of the dataset:

- They reveal "useful" causal dependencies.
- They can be used to distinguish between typical and abnormal behaviour.

# Frequent itemset mining in a nutshell

Attributes

| Transactions | # | $A$ | $B$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|---|---|
| | 1 | 0 | 1 | 1 | 0 | 0 |
| | 2 | 1 | 1 | 0 | 1 | 1 |
| | 3 | 1 | 0 | 0 | 1 | 0 |
| | 4 | 1 | 0 | 1 | 1 | 0 |
| | 5 | 1 | 1 | 0 | 0 | 1 |

Database $\mathcal{D}$ consists of $m$ transactions that can contain up to $n$ attributes.

A transaction $\mathcal{D}[i]$ can be viewed as a subset of attributes or a binary vector.

For example $\mathcal{D}[1] = \{B, C\}$, $\mathcal{D}[2] = \{A, B, D, E\}$ and $\mathcal{D}[5] = \{A, B, E\}$.

# Frequent itemset mining in a nutshell

| # | $A$ | $B$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 1 | 0 |
| 5 | 1 | 1 | 0 | 0 | 1 |

Frequencies are just normalised versions of supports:

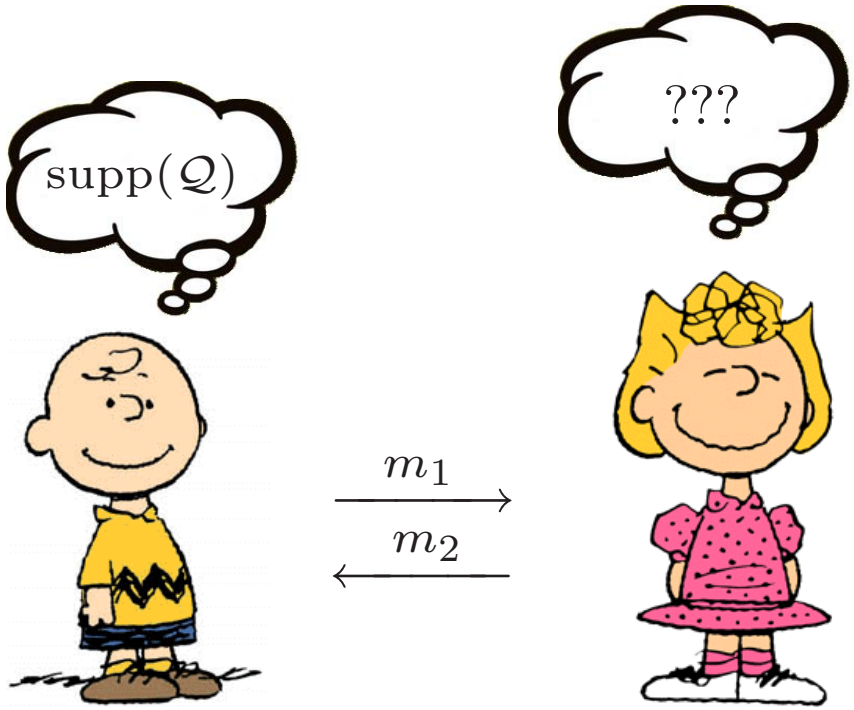
$$\mathrm{supp}(\mathcal{Q}) = \sharp\{i : \mathcal{Q} \subseteq \mathcal{D}[i]\} \quad \Rightarrow \quad \mathrm{supp}(\{A, D\}) = 3$$

$$\mathrm{freq}(\mathcal{Q}) = \frac{\mathrm{supp}(\mathcal{Q})}{m} \quad \Rightarrow \quad \mathrm{freq}(\{A, D\}) = 0.6$$

# Short outline of my talk

- Frequent itemset mining
- Private itemset support counting
- Previous work
- Online algorithms
- Offline algorithms
- Theoretical bounds
- Loopholes in the bounds

# Private itemset support counting




| # | $A$ | $B$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | 0 | 1 | 1 |
| 3 | 1 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 1 | 0 |
| 5 | 1 | 1 | 0 | 0 | 1 |

Client should learn $\mathrm{supp}(\mathcal{Q})$ while Server should learn nothing new.

## Previous work and possible applications

Privacy-preserving support counting techniques are also used when

- the database is divided horizontally between two or more parties;
- the database is divided vertically between two or more parties.

Then the itemset $\mathcal{Q}$ is a public and the main problem is secure aggregation.

**Our setting is different:**

- Clients can execute few queries, instead of buying the whole dataset.
- Server does not have to reveal the whole data.
- Client can do focused frequent itemset mining.

# Homomorphic encryption as a main tool

- Given a public key pk, one can compute an encryption $E_{\mathsf{pk}}(x)$ for $x \in \mathcal{P}$.
- Given a secret key sk, one can decrypt $D_{\mathsf{sk}}(E_{\mathsf{pk}}(x)) = x$.
- Without a secret key sk it is infeasible to distinguish $E_{\mathsf{pk}}(x)$ from $E_{\mathsf{pk}}(y)$.
- It is possible to compute with encrypted values

$$E_{\mathsf{pk}}(x) \cdot E_{\mathsf{pk}}(y) = E_{\mathsf{pk}}(x + y)$$
$$E_{\mathsf{pk}}(x)^y = E_{\mathsf{pk}}(x \cdot y)$$

  where computations are done in $\mathcal{P} \simeq \mathbb{Z}_n$.

# Efficient subset inclusion test

The duality between sets and binary vectors yields

$$\mathcal{Q} \subseteq \mathcal{D}[i] \iff \Big[\forall j:\ \mathcal{D}[i][j] = 0 \Rightarrow \mathcal{Q}[j] = 0\Big] \iff \sum_{j:\mathcal{D}[i][j]=0} \mathcal{Q}[j] = 0$$

Homomorphic PSI protocol

- Client sends encryptions $E_{\mathsf{pk}}(\mathcal{Q}[1]), \ldots, E_{\mathsf{pk}}(\mathcal{Q}[n])$.
- Server computes $d = \prod_{j:\mathcal{D}[i][j]=0} E_{\mathsf{pk}}(\mathcal{Q}[j])$ and replies $c = d^s,\ s \leftarrow \mathcal{P}$.
- Client computes $t = D_{\mathsf{sk}}(c)$. If $t = 0$ then $\mathcal{Q} \subseteq \mathcal{D}[i]$.

In the corresponding private itemset counting protocol, Server computes replays for every row $\mathcal{D}[i]$ and shuffles the answers before sending.

# Main properties of online PISC protocol

- Tolerable computational complexity. Server's workload is $\Theta(w+m)$ operations where $w$ is the number of ones in the database. Client's workload is $\Theta(n+m)$. This is feasible in practice.
- Heavy network traffic. The communication complexity is $\Theta(m)$. Roughly $320-2048$ bits per row.
- Server cannot do any pre-computations.

# Can we do better?

Alternatively, Server could precompute the supports of all itemsets.

- This is clearly infeasible as there are $2^n$ possible itemsets.
- However, it possible to compute the supports of all frequent itemsets, i.e. supports for all sets $X$ such that $\mathrm{supp}(X) \geq \tau$.

But then the client must obliviously access the database of frequent items.

- Such oblivious keyword transfer protocols exist.
- Some of them are quite efficient.
- No information is given for infrequent sets.

# Optimal solution for static databases

- Server encrypts database elements with pseudorandom keys

| | | | | |
|---|---|---|---|---|
| $0^\ell \Vert \operatorname{supp}(X_1)$ | $0^\ell \Vert \operatorname{supp}(X_2)$ | $0^\ell \Vert \operatorname{supp}(X_3)$ | $\ldots$ | $0^\ell \Vert \operatorname{supp}(X_k)$ |
| $\oplus$ | $\oplus$ | $\oplus$ | $\ldots$ | $\oplus$ |
| $\mathsf{ObPrf}_{\mathsf{sk}}(X_1)$ | $\mathsf{ObPrf}_{\mathsf{sk}}(X_2)$ | $\mathsf{ObPrf}_{\mathsf{sk}}(X_3)$ | $\ldots$ | $\mathsf{ObPrf}_{\mathsf{sk}}(X_k)$ |

  and sends it to Client.

- For each query $\mathcal{Q}$, Client and Server securely compute $\mathsf{ObPrf}_{\mathsf{sk}}(\mathcal{Q})$.
- Client decrypts a database element whenever it is possible.
- Online workload is poly-logarithmic for Client and Server!
- Initial communication complexity is tolerable.
- Not many good candidates for ObPrf are known.

# Theoretical bounds

Any exact PISC protocol gives a rise to PIR protocol and vise versa.

- We can encode $2^n$ bits information into $2^n \times n$ PISC database.
- The corresponding PIR$\leftrightarrow$PISC reduction is quite tight:
  - Any PISC protocol with sub-linear communication gives a raise to PIR protocol with sub-linear communication ("a challenging puzzle for cryptographers").
  - At any time the discrepancy between best communication complexities is different by a logarithmic term.
- Radically new ideas are needed to reduce the communication complexity of PISC protocols.

## How can we beat these theoretical bounds?

Random sampling from the database provides good approximations:

- Communication complexity is independent from number of rows.
  - Roughly $40,000$ rows are needed to approximate frequencies with absolute precision $0.01$ with the confidence level $99.9\%$.
- Hoeffding and Chernoff bounds provide exponentially small failure probabilities w.r.t. number of rows.
- More complex bounds based on VC-dimension give failure bounds for simultaneous approximation of all subset frequencies.

# Conclusions

- Private itemset counting is a hard problem. Any significant advance gives a raise to new PIR protocol.
- We need good oblivious keyword transfer protocols.
  - Low communication complexity of such protocols is over-hyped.
  - Actually, we need protocols with poly-logarithmic online complexity.
  - Offline complexity is quite irrelevant.
- Randomised methods can provide loopholes to beat theoretical bounds.
- State of the art cryptography cannot support more advanced representations of frequent itemsets like condensed sets, FP-trees etc.