MTAT.07.006 Research Seminar in Cryptography

Building a secure aggregation database

Dan Bogdanov
University of Tartu
db@ut.ee

What this talk is about

- Privacy concerns in data analysis
- What can we do to preserve the privacy of data donors
- Our progress in building a privacy preserving data aggregation engine

What is considered private/sensitive?

If we made a survey and asked people about their

- health information
- political preferences
- sexual behaviour

then the answers will be considered **sensitive data**.

What are we protecting? From who?

Since the data is private to an entity (a person or an organization), we have to protect the answers from everyone but their author..

More specifically - we want to keep all participants except for the donor from seeing his or her set of answers.

But we also need some entity (a data miner), which could provide us with statistical results on the data without seeing ”too much” of it. There can be more than one miner.

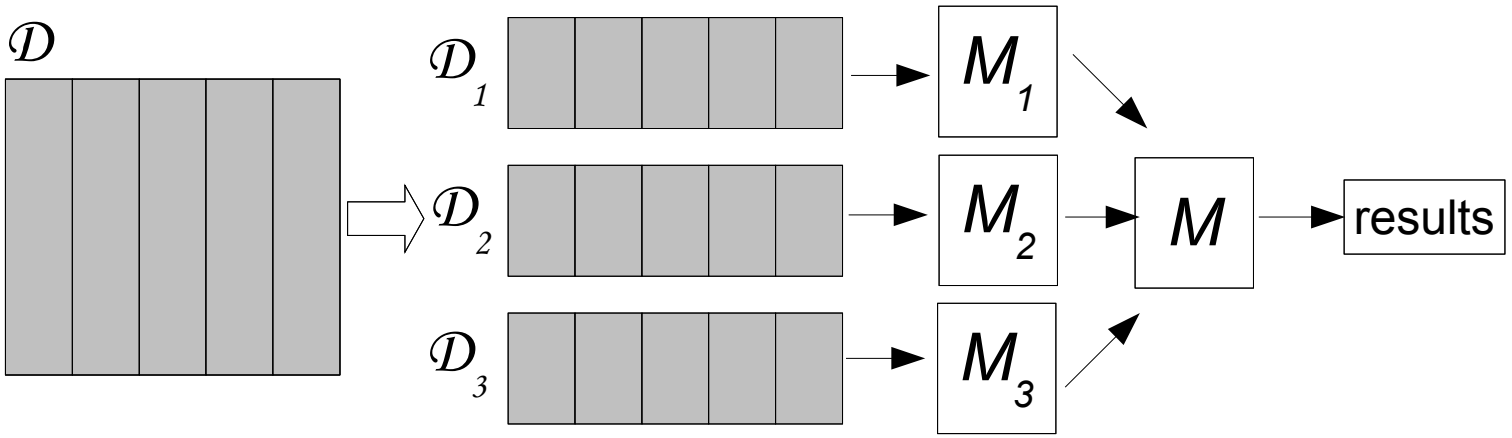
Our setup

- Let $W_1, \ldots, W_n$ be the participants who provide us with the data.
- Each participant answers m questions
- The database $\mathcal{D}$ is a $n \times m$ matrix of answer values.

For the current example, let's say that each participant gives us one row of the database.

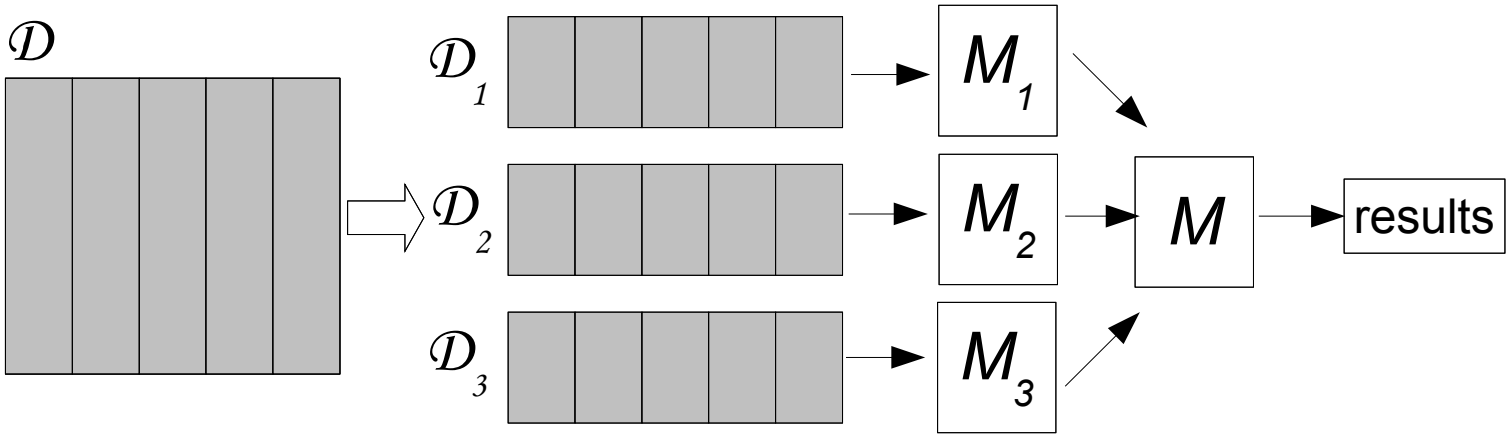
Idea 1: Distribute the rows

We have multiple data miners. Each one gains access to a subset of rows and calculates statistics on these rows.



In the example, $\mathcal{D}$ is the database. $\mathcal{D}_1, \mathcal{D}_2$ and $\mathcal{D}_3$ are subsets of the database. M_1, M_2 and M_3 are data miners working on the subsets and M is the master miner, who combines the results of the miners.

Idea 1: Distribute the rows - verdict

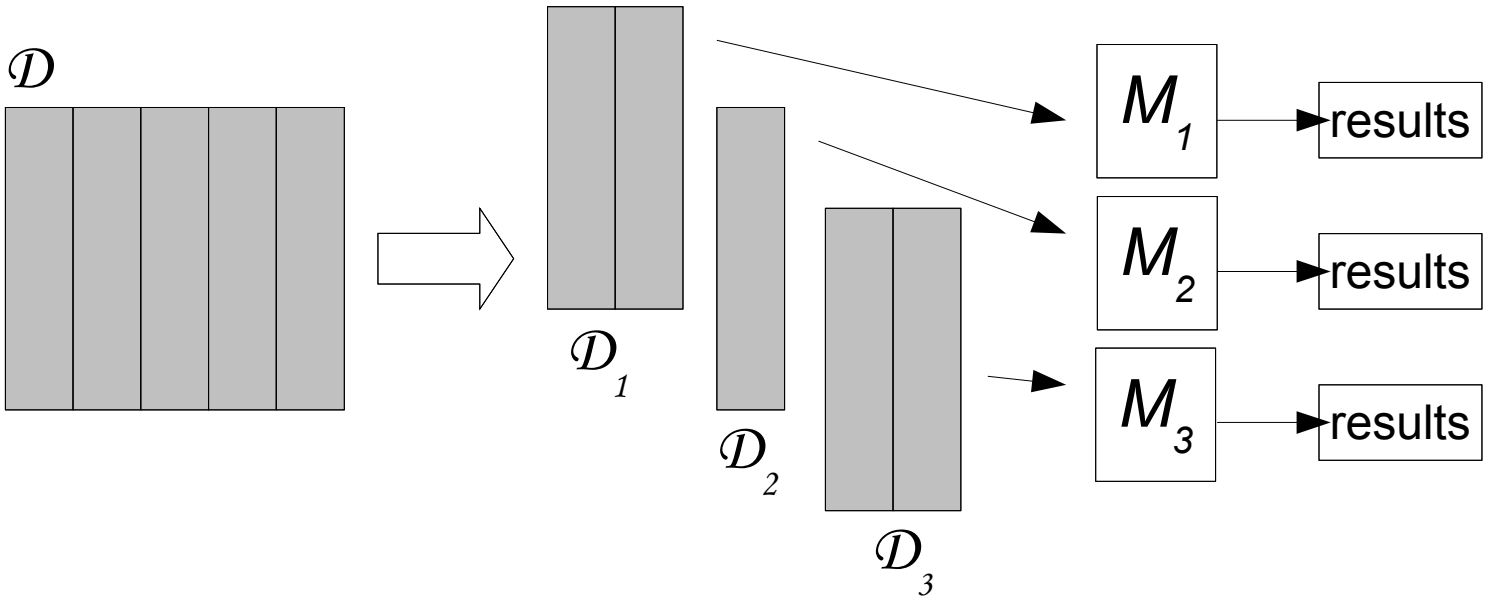


This method is not private - a miner can see complete information about a participant. Not every aggregation algorithm is easily distributable to requirements of this method

Method is not secure and therefore not acceptable.

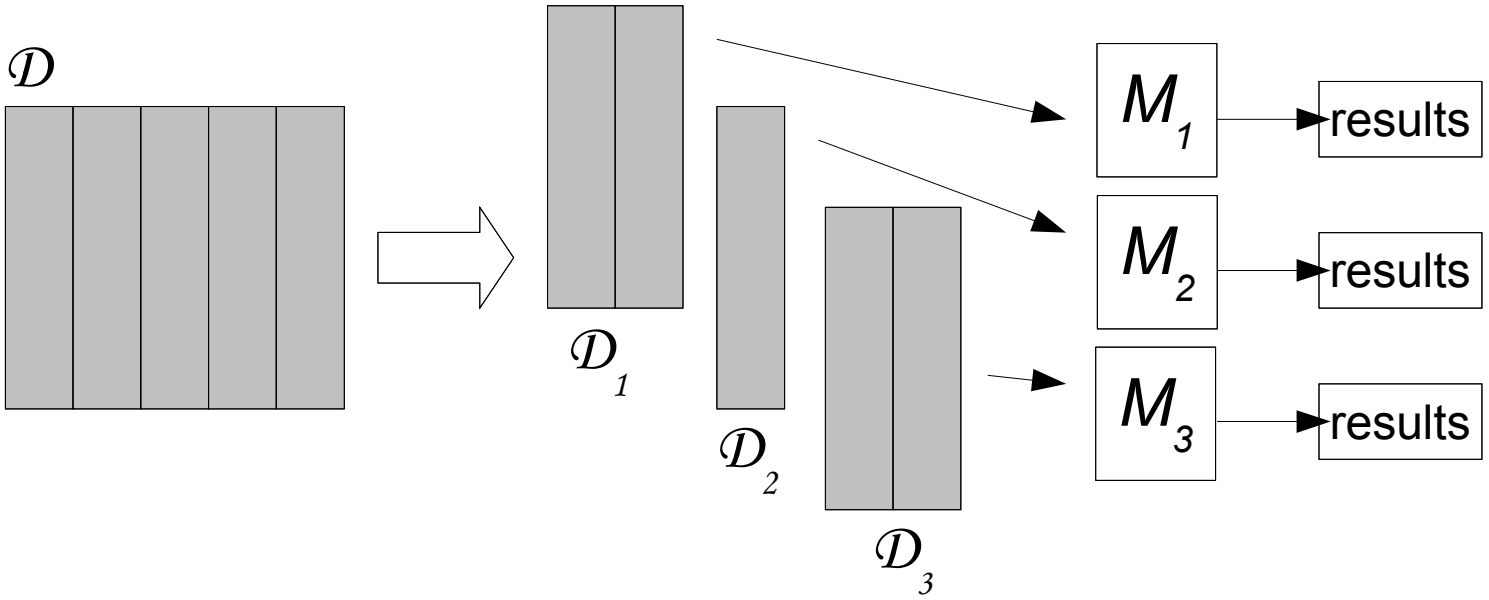
Idea 2: Distribute the columns

We have multiple data miners. Each one gains access to a subset of columns and calculates statistics based on these columns.



In this example $\mathcal{D}_1, \mathcal{D}_2$ and $\mathcal{D}_3$ are column subsets of $\mathcal{D}$. M_1, M_2 and M_3 are data miners working on the subsets.

Idea 2: Distribute the columns - verdict

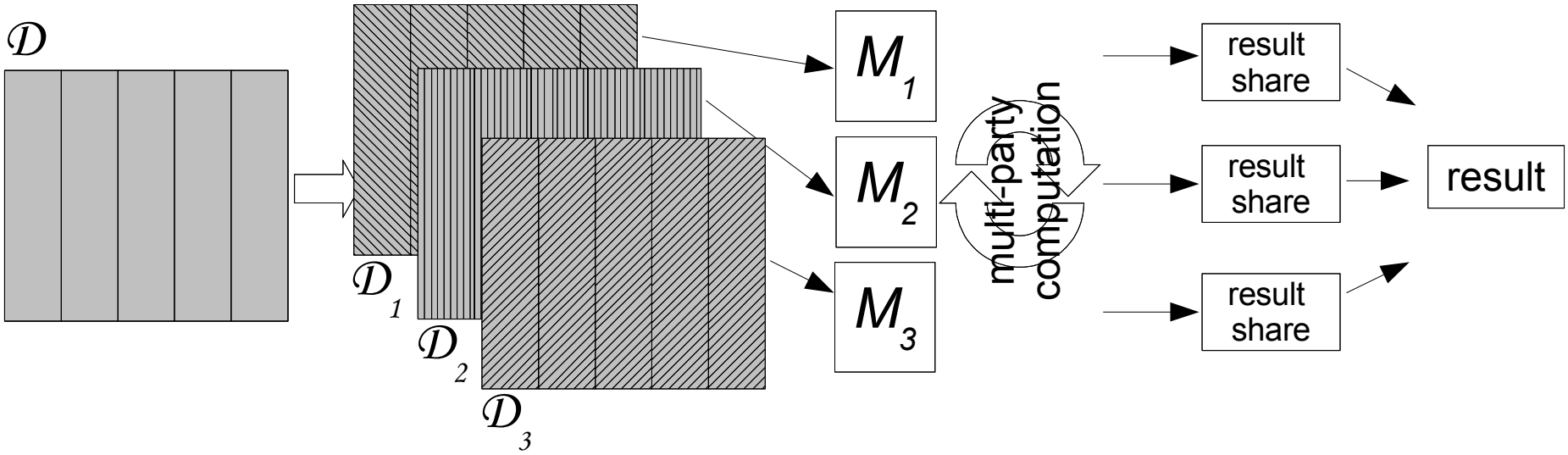


This method is not private - some miner may have enough attributes to access personal data. Since the miner can use only some attributes, its analysis capabilities are crippled.

Method is not secure and therefore not acceptable.

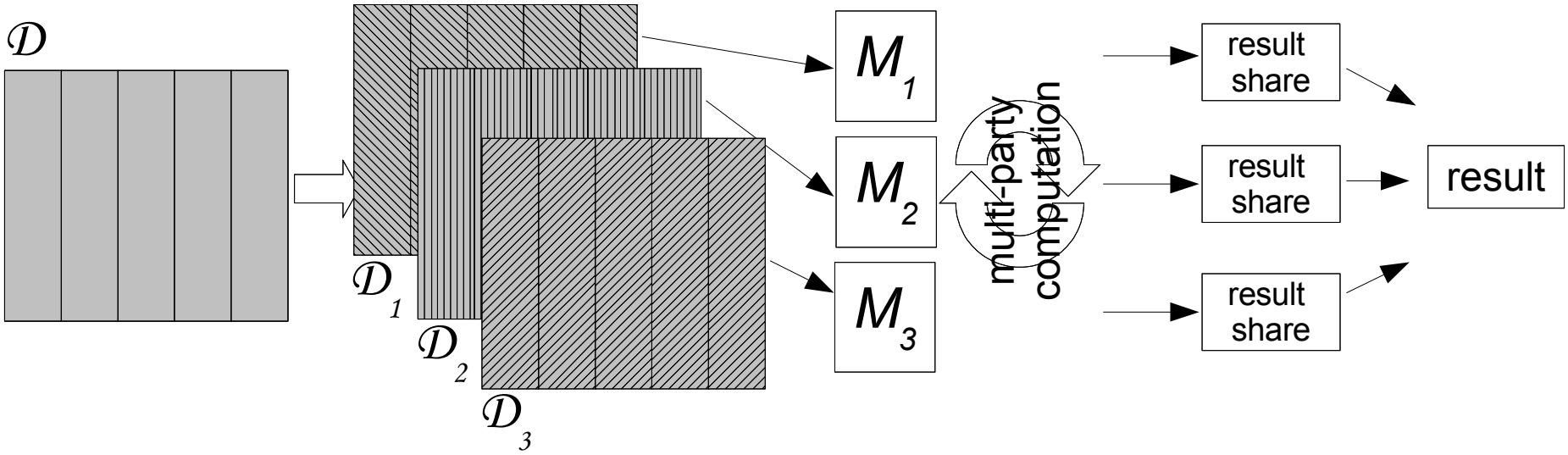
Idea 3: Distribute the values

We have multiple data miners. Each one is given a part of all the values. This is done by using **secret sharing**. The miners use **secure multi-party computation** to calculate aggregations.



$\mathcal{D}_1, \mathcal{D}_2$ and $\mathcal{D}_3$ contain shares of values in $\mathcal{D}$. M_1, M_2 and M_3 use multi-party calculation to calculate result shares.

Idea 3: Distribute the values - verdict



This method can be proven secure. However, most of the calculations and other operations require specific protocols, which can be time-consuming.

Method is secure, but is it feasible?

Secure multi-party computation in a nutshell

- Assume that we have n participants $p_1, \ldots, p_n$.
- Each participant i knows an input value x_i.
- We calculate a function $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$
- Each node i will get the value of y_i and nothing else.

Secret sharing in a nutshell

- Assume that we have an input value s that we wish to keep secret.
- We have n nodes available for computation.
- We take s as the input and output n bitstrings $s_1, \ldots, s_n$ (shares).
- The value s can be reconstructed only if all shares are available.

Building a secure aggregation database

- Our setup consists of three miners and any number of participants.
- Secret sharing is used to distribute the database between miners.
- Multi-party computation is used to calculate aggregations.
- Basically the miners make up a distributed processor

Properties of the mining engine

The distributed mining engine makes use of a number of facilities. Each miner has the following components:

1. Persistent storage — the database $\mathcal{D}$
2. Run-time storage — the heap $\mathcal{H}$ and the stack $\mathcal{S}$
3. Instruction scheduler for processing incoming commands
4. Network messaging for running protocols and exchanging data

How do the miners operate — instruction processing

- A participant running the control program connects to the miners.
- The controller library processes the function calls in the program.
- The resulting commands and the parameters are sent to the miners.
- The commands are received and processed by the miners.

Storing shares in the miner's database

- The controller calculates shares $s_1, \ldots, s_n$ of the input value.
- The triplet (x, y, s_i) is sent to to the miner M_i.
- Each miner stores it in its database: $\mathcal{D}_{i_{xy}} = s_i$

Share multiplication

- We want to calculate the product of two values in the database.
- Let the first value be $x = \mathcal{D}_{kl}$ and the second one $y = \mathcal{D}_{mn}$.
- We ask each miner M_i to multiply $\mathcal{D}_{i_{kl}}$ and $\mathcal{D}_{i_{mn}}$.
- The miners run the protocol and get the shares of the product xy.

Implementation plan

- First milestone — a library for fast application prototyping.
- Second milestone — an optimised version of the database.

Implemented operations

As of November 2006 we have implemented the following instructions:

1. Requesting that the miner loads a database with the given name
2. Storing shares in the miner's database
3. Multiplying two shares in the database

End of talk

Thanks for listening!