# Confidential Truth Finding with Multi-Party Computation

**Angelo Saadeh**[1,6], Pierre Senellart[1,2,4,5,7], Stéphane Bressan[1,3,7]

[1]CNRS@CREATE LTD, Singapore

[2]DI ENS, ENS, PSL University, CNRS, Paris, France

[3]National University of Singapore, Singapore

[4]Inria, Paris, France

[5]Institut Universitaire de France, Paris, France

[6]LTCI, Télécom Paris, IP Paris, Palaiseau, France

[7]IPAL, CNRS, Singapore, Singapore

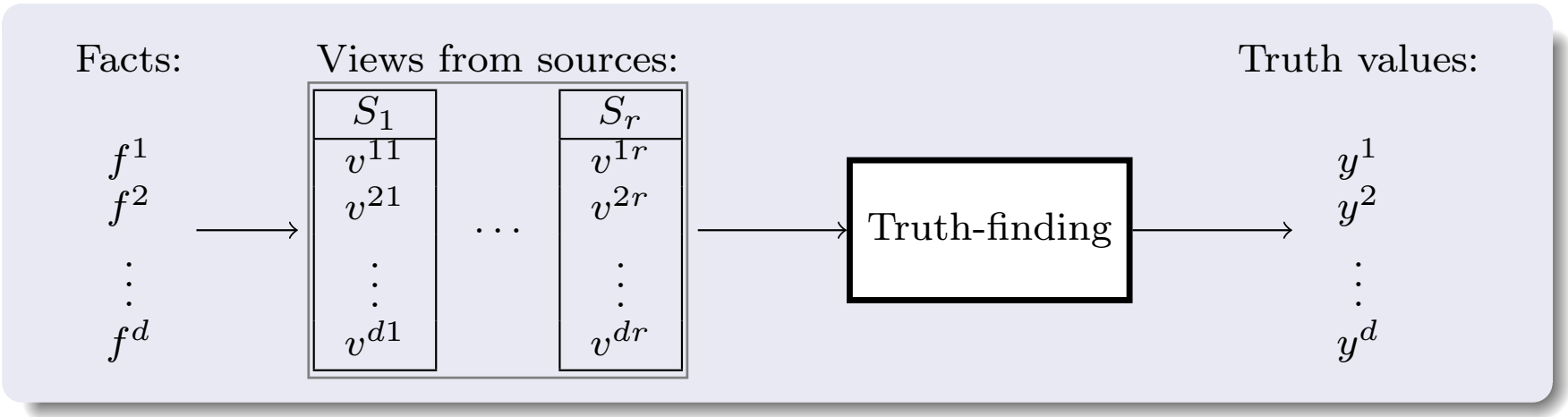
DEXA 2023, Penang, Malaysia, 28-30 August 2023

# Truth-finding with Disagreeing Sources

Let $f^1, \ldots, f^d$ be facts/queries.

Each of the $r$ sources inputs their view $(v^{ij})_{i=1\ldots d, j=1\ldots r} \in \{-1, 0, 1\}^{d \times r}$ of each fact.

Truth values $(y^i)_{i \in \{1,\ldots,d\}} \in [-1, 1]^d$

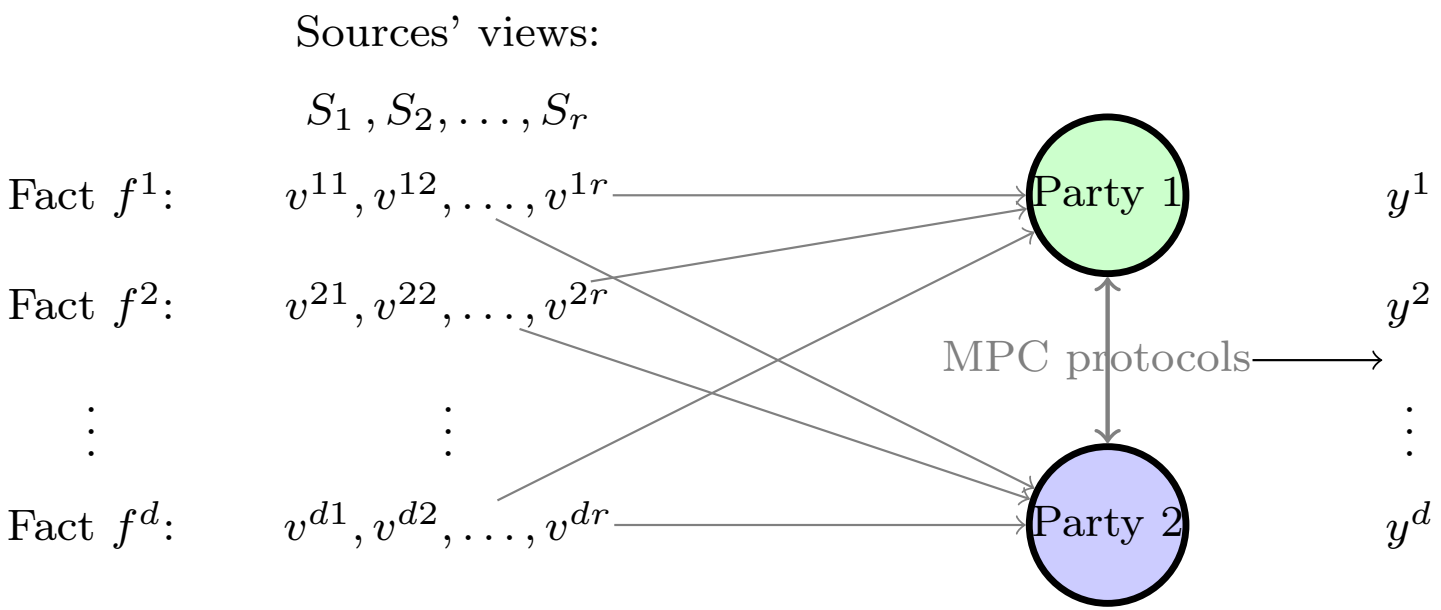



Truth-finding algorithms [YHY08] can be used to complete missing data.

# Truth-finding on Confidential Views

Using **MPC** in truth-finding algorithms protects the views of the sources.

Asking each of the $r$ sources for their view on $d$ facts.

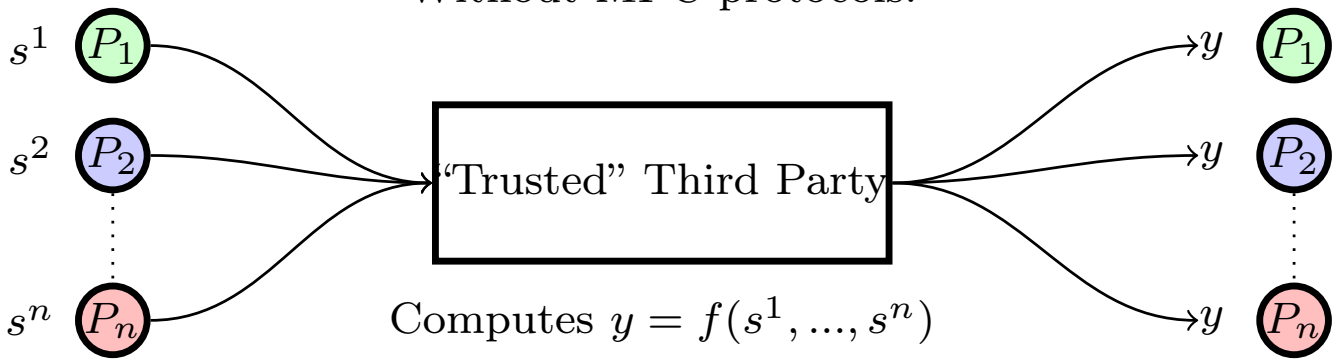



We compute $(y^i)_i \in [-1,1]^d$ using **MPC** to protect $(v^{ij})_{i,j} \in \{-1,0,1\}^{d \times r}$.

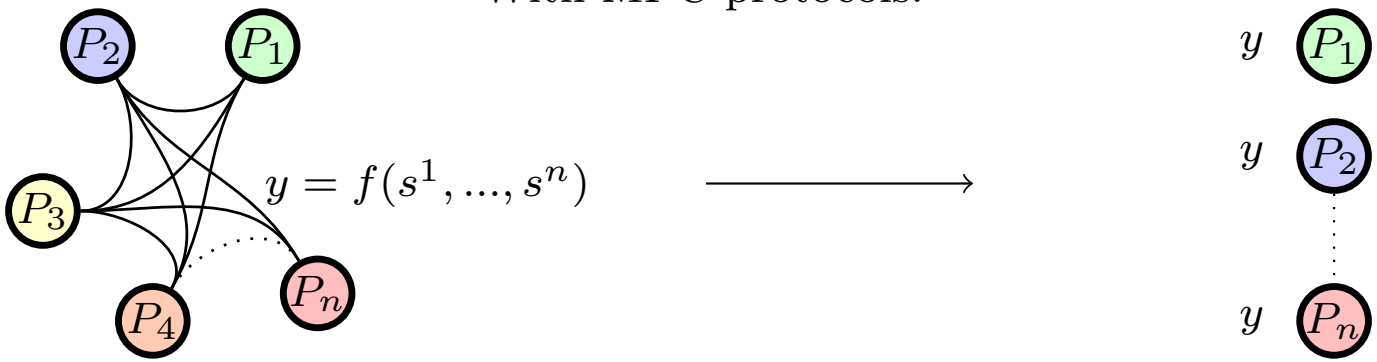
# Secure Multi-party Computation (MPC)

Let $f$ be a public function.

Without MPC protocols:




With MPC protocols:



# Two-party Additive Secret-sharing

Let $l \in \mathbb{N}^*$, $\mathbb{Z}/2^l\mathbb{Z}$ a finite ring.

**Two-party additive secret-sharing $\Pi_{\text{share}}$ [MGW87]**

**Input:** $P_1$ holds $s^1$

1. $P_1$ generates $s_2^1 \xleftarrow{\$} \mathbb{Z}/2^l\mathbb{Z}$
2. $s_1^1 \leftarrow s^1 - s_2^1 \mod 2^l$
3. $P_1$ sends $s_2^1$ to $P_2$

Notation: $[s^1]$ correspond to the shares $s_1^1$ and $s_2^1$ of $s^1$.

Computing mod $2^l$ is crucial to keep the shares uniformly distributed in the ring.

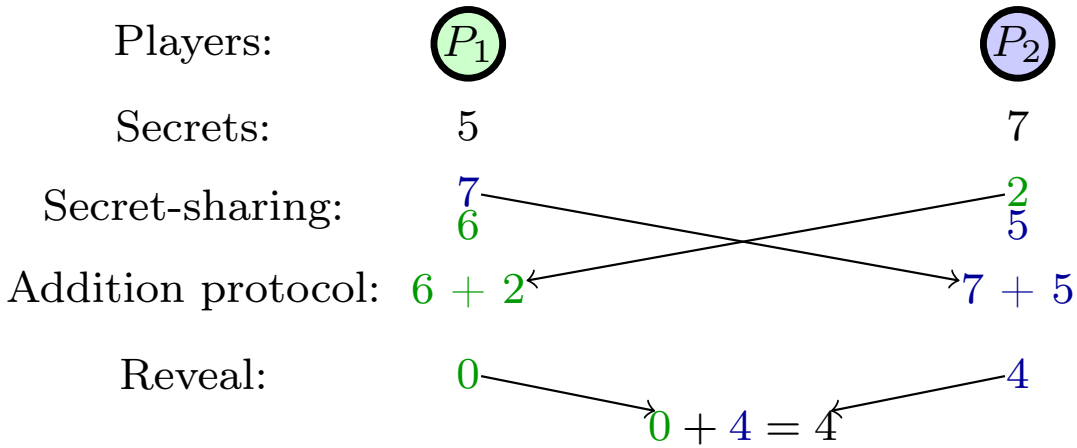
# Addition protocol

## Two-party addition protocol $\Pi_{\text{add}}$

**Input:** $P_i$ holds $x_i$,$y_i$ for $i$ in $\{1, 2\}$
**Output:** $P_i$ holds $z_i$ for $i$ in $\{1, 2\}$ such that $z_1 + z_2 = x + y$

1. $P_i$ computes $z_i \leftarrow x_i + y_i$ for $i$ in $\{1, 2\}$

**Example of the sum of two secrets modulo 8:**

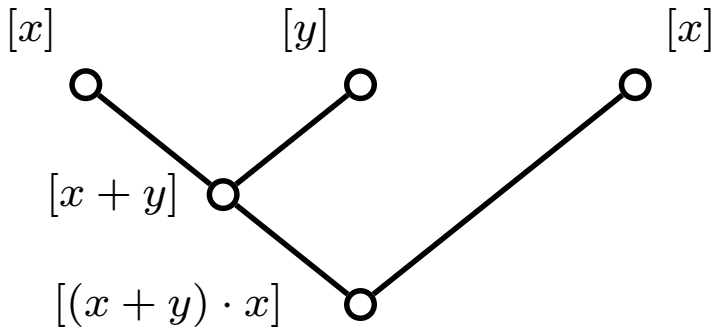


# Arithmetic Circuit Evaluation in MPC

With $(x_1, x_2) \leftarrow [x]$ and $(y_1, y_2) \leftarrow [y]$ we can have:

- $(z_1, z_2) \leftarrow [x + y]$ with an addition protocol
- $(t_1, t_2) \leftarrow [xy]$ with a multiplication protocol [Bea91]

We can privately evaluate the arithmetic circuit of a function $f : (x, y) \rightarrow (x + y) \cdot x$ :



# From Real Elements to Finite Ring Elements [MZ17]

| Real elements | Decimals | Finite Ring elements |
|---|---|---|
| $I = [-2^{l-p-1}, 2^{l-p-1})$ | $D_{p,l} = \{k \cdot 2^{-p}, k \in \mathbb{Z}\}$ | $\mathbb{Z}_{2^l} = [0, 2^l) \cap \mathbb{Z}$ |

$$x \longrightarrow \tilde{x} \longrightarrow \overline{x}$$

$$\text{round}(x) = k_x 2^{-p} \qquad k_x \mod 2^l$$

**Example for $l = 2, p = 1$:**

| $I = [-1, 1)$ | $D_{1,2} = \{-1, -0.5, 0, 0.5\}$ | $\mathbb{Z}/2^2\mathbb{Z} = \{0, 1, 2, 3\}$ |
|---|---|---|
| $-1$ | $\widetilde{-1} = -1 = -2 \cdot 2^{-1}$ | $\overline{-1} = -2 \mod 2^2 = 2$ |
| $-0.5$ | $\widetilde{-0.5} = -0.5 = -1 \cdot 2^{-1}$ | $\overline{-0.5} = -1 \mod 2^2 = 3$ |
| $0.6$ | $\widetilde{0.6} = 0.5 = 1 \cdot 2^{-1}$ | $\overline{0.6} = 1 \mod 2^2 = 1$ |

For simplicity, we omit the bar.

# Computing Real Functions in the Finite Ring

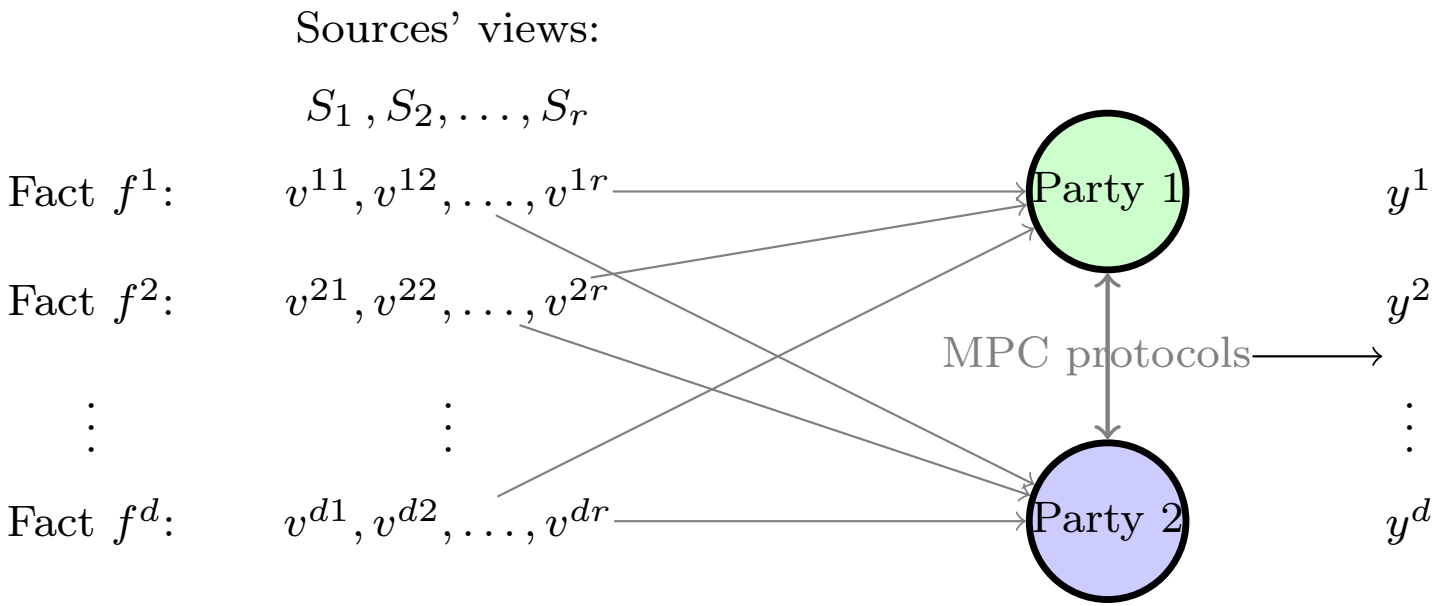
We compute the arithmetic circuits of real functions using $\Pi_{\text{add}}$ and $\Pi_{\text{mult}}$.

**Example:** For a positive secret $[x]$ compute $\left[\frac{1}{x}\right]$ in MPC [Kno+21].

1. Define the function $g(y) = x - \frac{1}{y}$
2. Use Newton-Rapshon iterations to find the root $x$ of $g$ because $g(\frac{1}{x}) = 0$
3. The sequence is defined as follows: $y_{n+1} = y_n^2 x + 2y_n$

# Truth-finding Security Model

- There are $d$ binary facts $f^1, \ldots, f^d$
- For $i \in \{1, ..., d\}, j \in \{1, ..., d\}$, source $j$ answers $v^{ij} \in \{-1, 0, 1\}$ to $f^i$
- A truth-finding algorithm returns a truth value $y^i \in [-1, 1]$ for each fact $f^i$



# A Truth-finding Algorithm: Cosine

## Simplified version of Cosine [Gal+10]

**Input:** The answers $(v^{ij})_{i,j}$
**Output:** The truth values $(y^i)_i$

1. Initialize truth values $(y^i) \leftarrow 1$
2. For a number of iterations do:
   1. For every source $j$:

$$\theta^j \leftarrow \frac{\sum_{i,v^{ij}=1} y^i - \sum_{i,v^{ij}=-1} y^i}{\sqrt{\left(\sum_{i,v^{ij}\neq 0} v^{ij}\right)\left(\sum_{i,v^{ij}\neq 0} (y^i)^2\right)}}$$

   2. For every fact $f^j$:

$$y^i \leftarrow \frac{\sum_{j,v^{ij}=1} (\theta^j)^3 - \sum_{j,v^{ij}=-1} (\theta^j)^3}{\sum_{j,v^{ij}\neq 0} (\theta^j)^3}$$

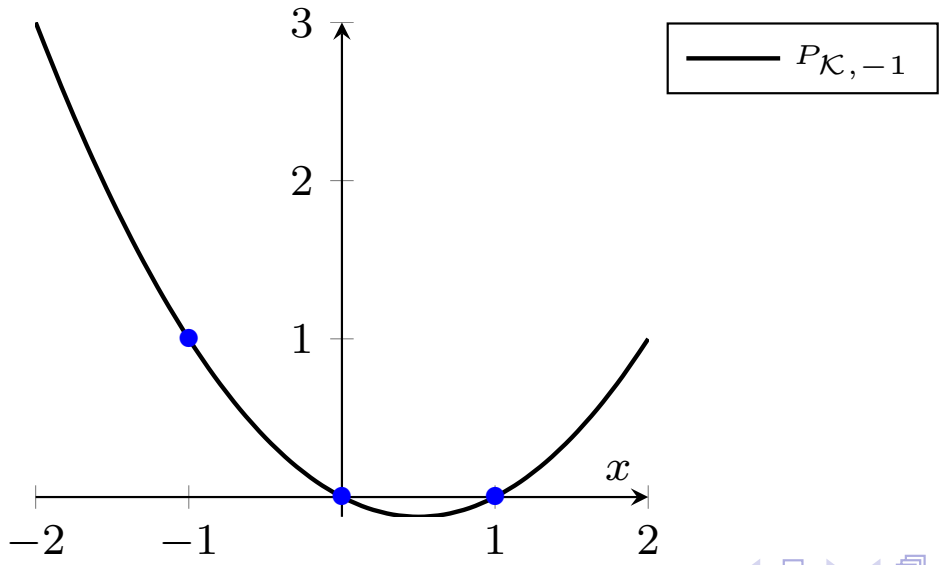
# Pseudo-equality Test with Polynomial Evaluation

**For a secret $[x]$ and a public element $k$ we need:**

$$[y] = \Pi_{\text{equal}}([x], k) \text{ with } y = \begin{cases} 1 & \text{if } x = k \\ 0 & \text{otherwise.} \end{cases}$$

For truth-finding algorithms, $k \in \mathcal{K} = \{-1, 0, 1\}$.
A classic equality test could be replaced by a degree-two polynomial $P_{\mathcal{K},k}$.

**Examples for** $k = -1, P_{\mathcal{K},-1}(x) = \frac{1}{2}(x^2 - x)$**:**



# Alternative for the Secure Inverse Algorithm for Negative Values

**Secure inverse algorithm [Kno+21]:**

For a secret $[x]$
If $x > 0$, the inverse is computed as:

$$[y] = \Pi_{\text{inv}}([x]) \text{ with } y = \frac{1}{x}$$

If $x < 0$, the inverse is computed as:

$$[y] = \Pi_{\text{sign}}([x]) \cdot \Pi_{\text{inv}}(\Pi_{\text{abs}}([x])) \text{ with } y = \frac{\text{sign}(x)}{|x|} = \frac{1}{x}$$

If $x < 0$, we replace the sign computation with two multiplications:

$$[y] = [x] \cdot \Pi_{\text{inv}}([x] \cdot [x]) \text{ with } y = \frac{x}{x^2} = \frac{1}{x}$$

# Results on Confidential Truth-finding

**Cosine on MNIST**
120 **facts and** 15 **sources:**

| | Non-confidential | MPC with classic inverse | MPC with optimized inverse |
|---|---|---|---|
| Wall time | $10^{-4}$ s | 0.47 s | 0.44 s |
| Accuracy | 90% | 90% | 90% |

Related solutions:

| **Reference** | **Algorithms** |
|---|---|
| [Chi+16; NBK15] | Majority Voting |
| [Mia+15; Zhe+20; ZDW18] | Conflict Resolution on Heterogeneous Data (CRH) [Li+16] |
| **[SSB23]** | Cosine and 3-Estimates [Gal+10] |

# Conclusion:

**Take home message:**

Confidential truth-finding could be achieved with secret-sharing-based MPC

**Contributions of the paper:**

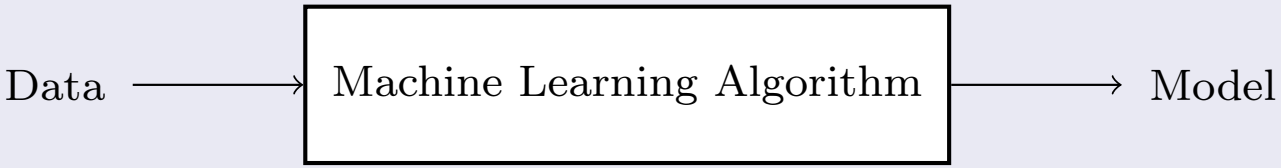
- MPC primitives for functions used in Truth-finding
- Arithmetic MPC protocols for the equality tests on finite sets

**Future research:**

- Truth-finding with differential privacy
- Truth-finding algorithm that protects the facts

Questions.

# Machine Learning and Data




**Examples of federated learning:**

Data from banks → Fraud detection

Data from social media and eCommerce platforms → Marketing segmentation

Data from cars → Vehicle automation

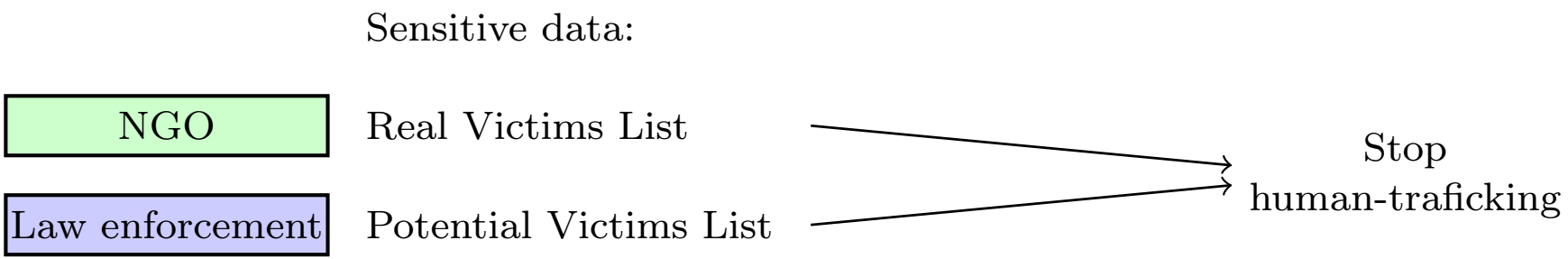
# Federated Learning with Confidential Data

**Constraints:**

- Data is too confidential to be shared.
- Rules and regulations prevent sharing sensitive data.

**Secure Multi-party Computation (MPC)** allows computing the model output without revealing any participant's secret data input.

**Example of MPC to fight human trafficking by Roseman Labs:**



# Truth Finding

- Truth-finding algorithms aim to know if a statement is correct or not
- They involve collecting sources answers to queries, and analyzing the answers
- These algorithms could be used to complete missing data

Given the answers to $d$ queries from $r$ sources: $(v^{ij})_{i=1\ldots r, j=1\ldots d}$

The algorithm outputs $y^1, \ldots, y^d$ the truth value of each of the $d$ queries.

# MPC-friendly Normalization Alternative

**For a vector of secrets $([x^1], \ldots, [x^n])$ we need:**

$$([y^1], \ldots, [y^n]) = \Pi_{\text{norm}}([x^1], \ldots, [x^n]) \text{ with } y^i = \frac{x^i - \min_i x^i}{\max_i x^i - \min_i x^i}.$$

The goal is to scale the elements of the secret vector to $[0, 1]$ with less communication.

We apply a linear transformation $h(x) = 0.5x + 0.25$ instead of $\Pi_{\text{norm}}$: