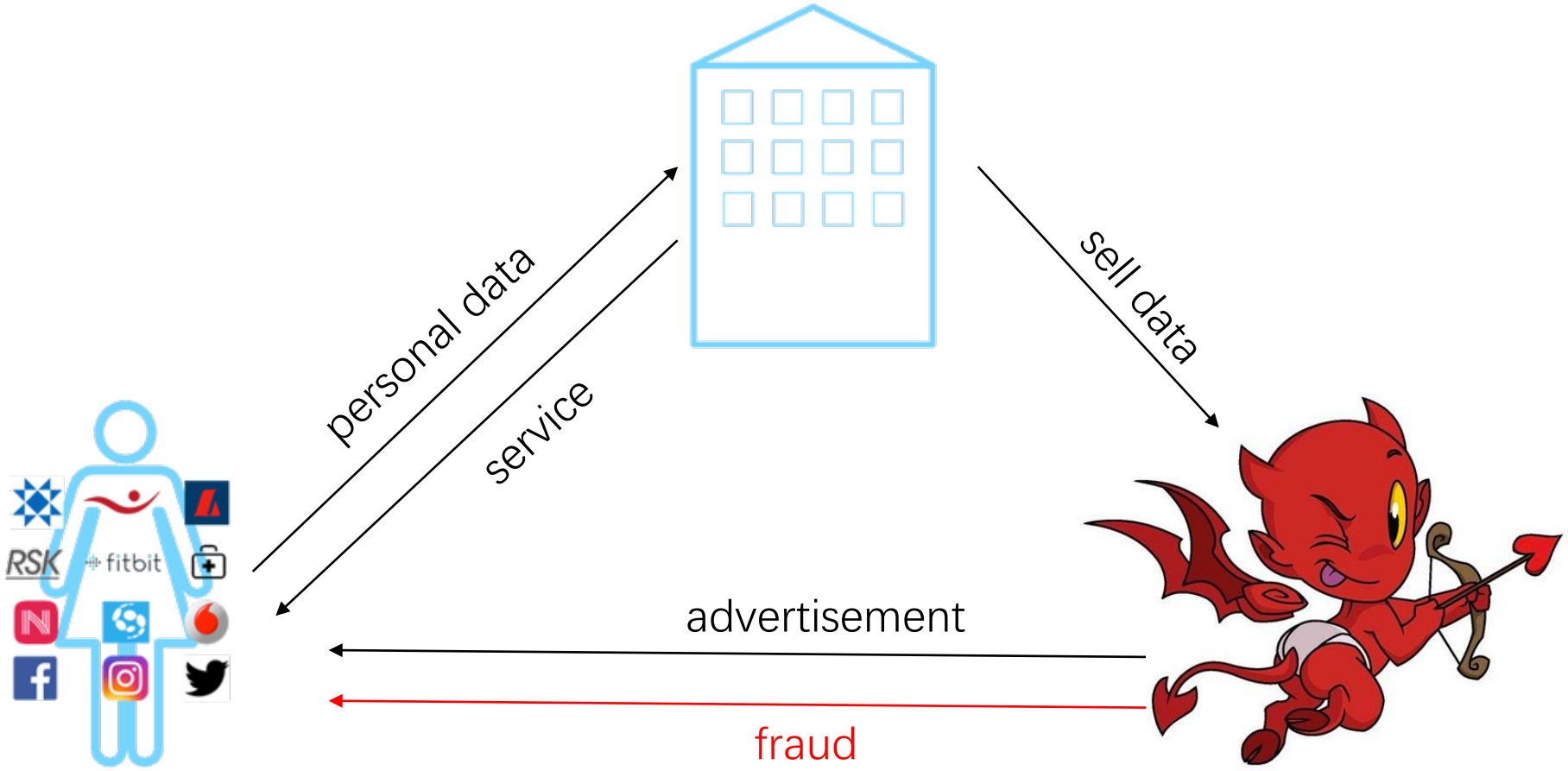
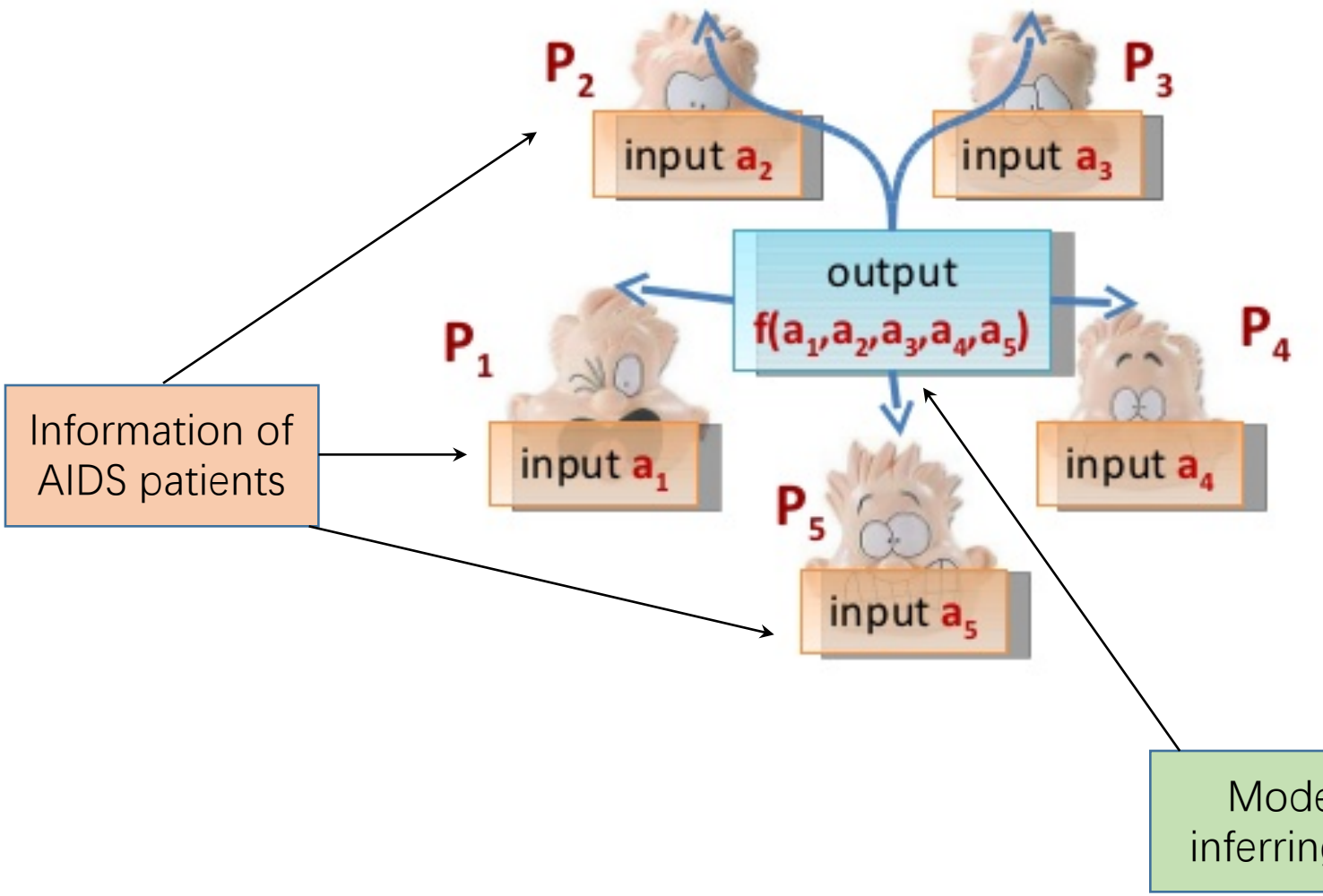
PEM: A Practical Differentially Private System for Large-Scale Cross-Institutional Data Mining

Yi Li*, Yitao Duan†, Wei Xu*
* Institute for Interdisciplinary Information Sciences, Tsinghua University
† Netease Youdao.

Privacy: An Important Issue



Privacy-Preserving Data Mining



- Medical research
- Online education
- Finance
- .....

Existing Approaches

- **Cryptographic methods are inefficient**
  - Yao Garbled Circuit
  - Homomorphic Encryption

- **Existing general noise-based methods add too much noise**
  - each client adds noise to its input

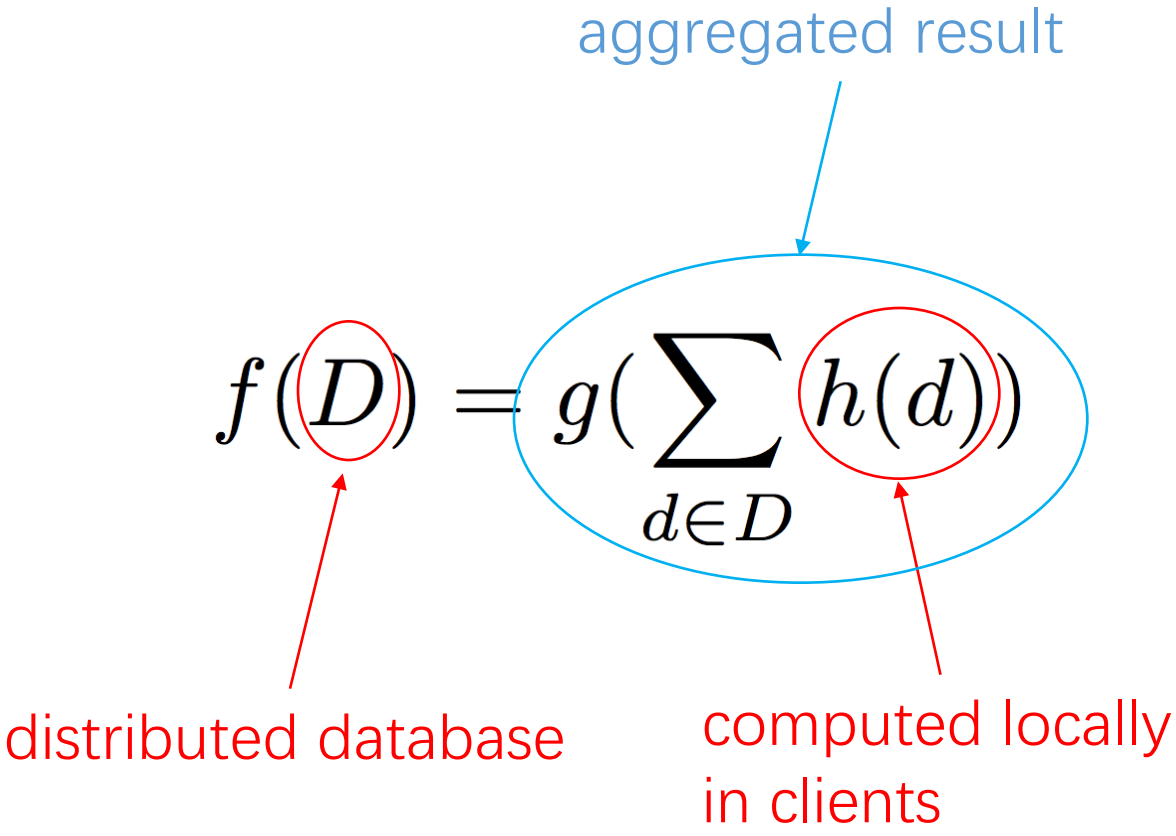
- **Noiseless methods make too many assumptions**
  - algorithm-specific
  - assuming too much about the datasets

PEM: A Practical Tradeoff Among Privacy, Utility and Overhead

Practicability

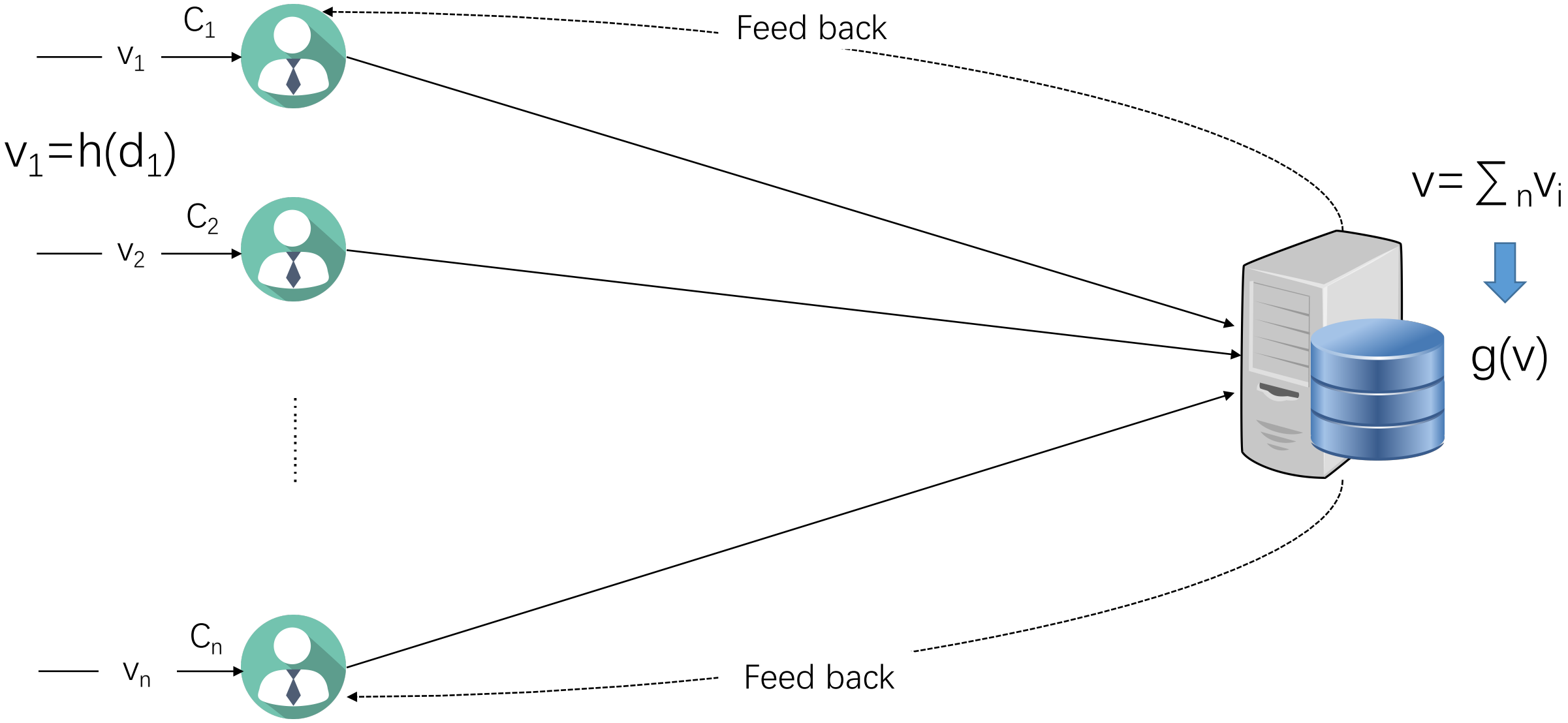
- Accuracy: **reduce the amount of noise significantly**
- Privacy: **achieve provable privacy**
- Convenience: **provide complete solution**
- Generality: **support common machine learning algorithms (e.g. LR, k-means, Apriori)**

Problem Formulation

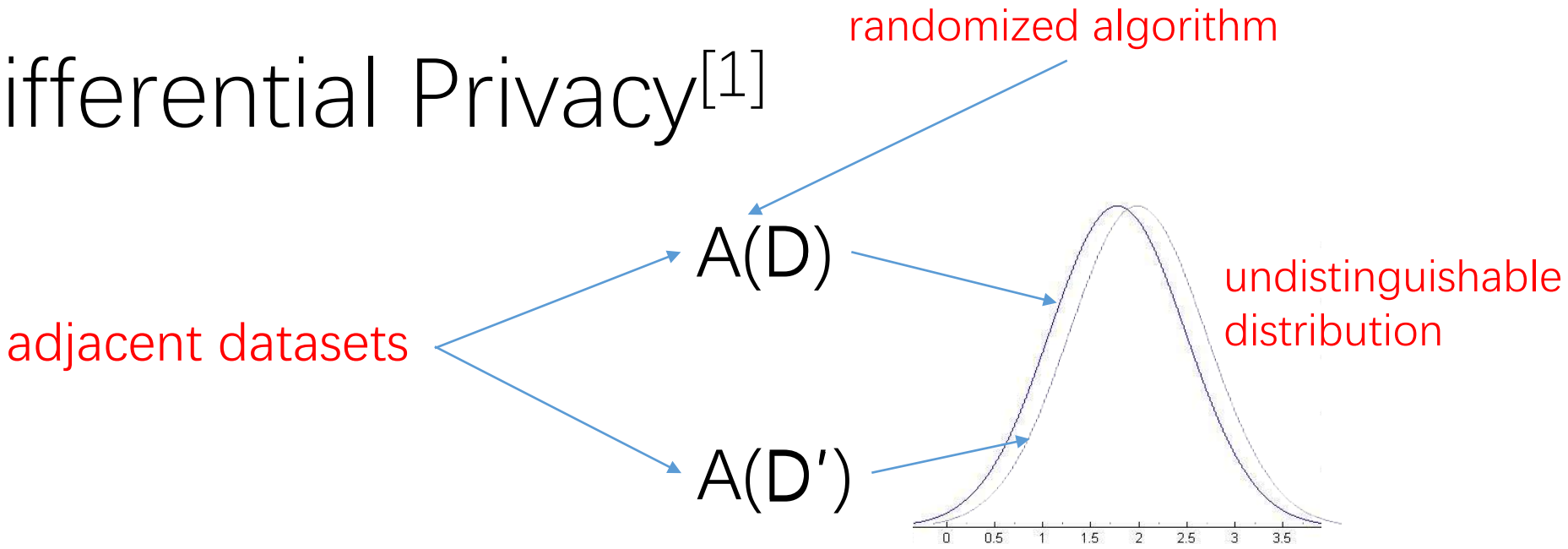


gradient descent, k-means, Apriori, SVD, EM, …

Problem Formulation



Differential Privacy[1]

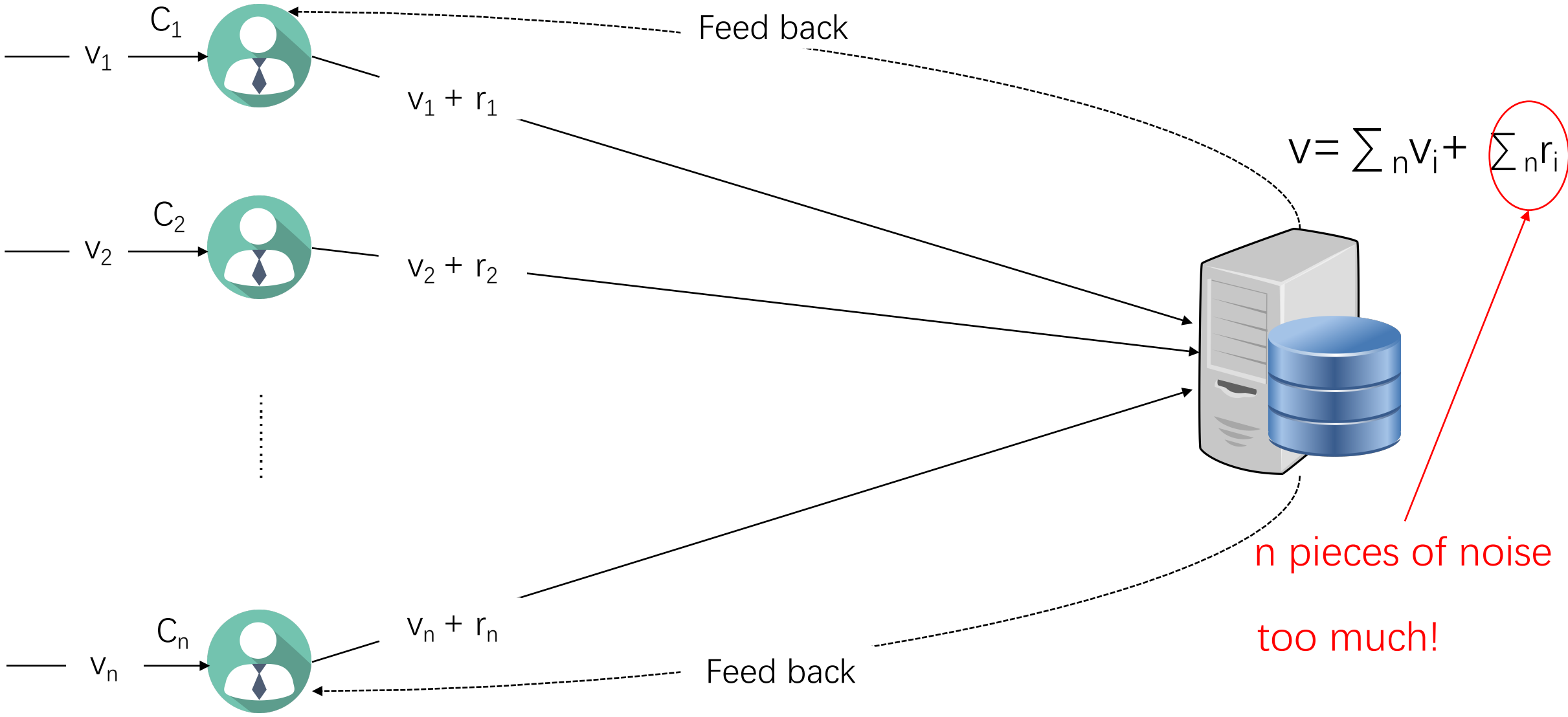


ε-differential privacy: smaller ε means stronger privacy

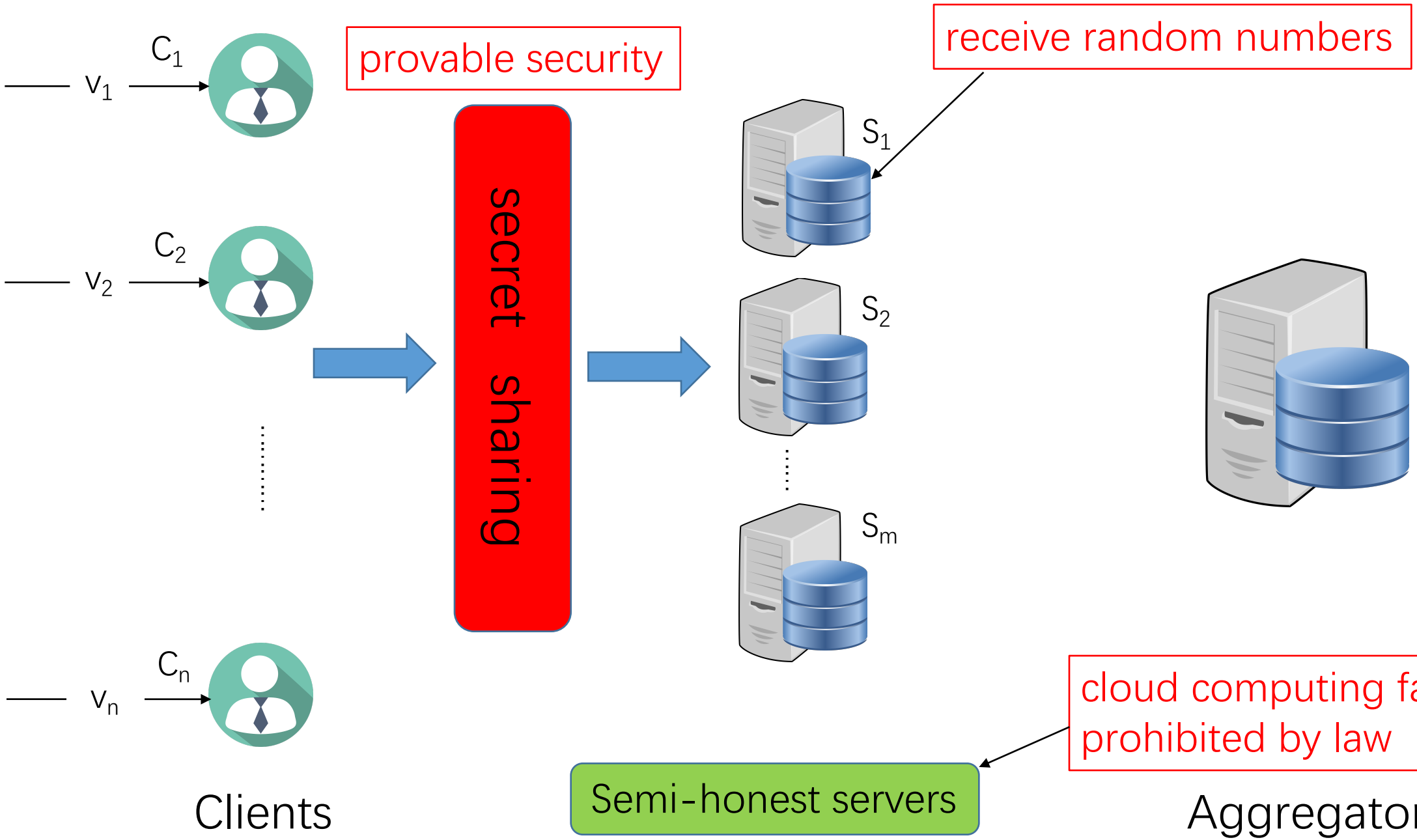
How to? Add noise Laplace distribution

[1] Dwork, Cynthia. ” Differential privacy.” InICALP, pages 1–12, 2006.

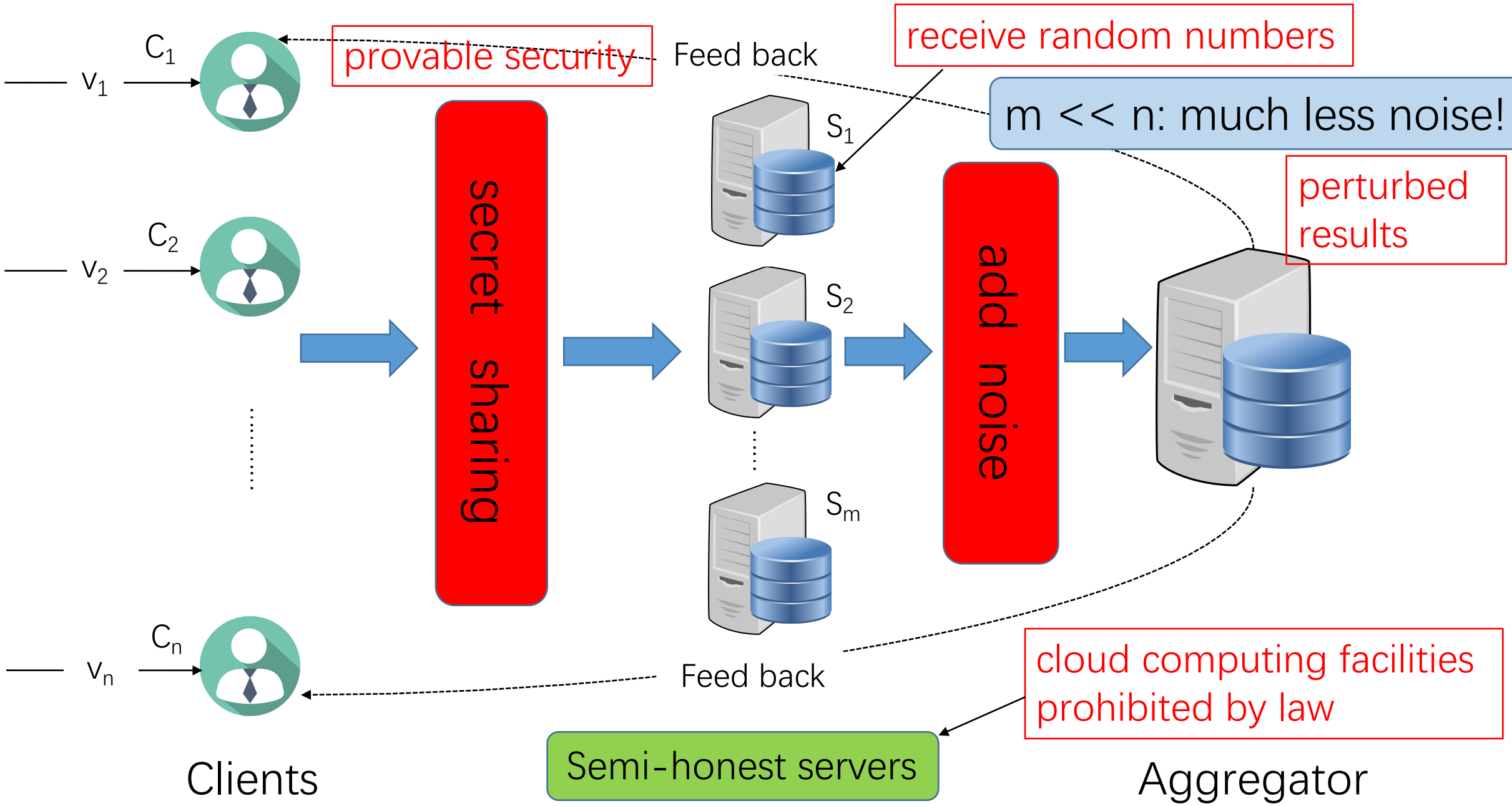
Noise-Only Method



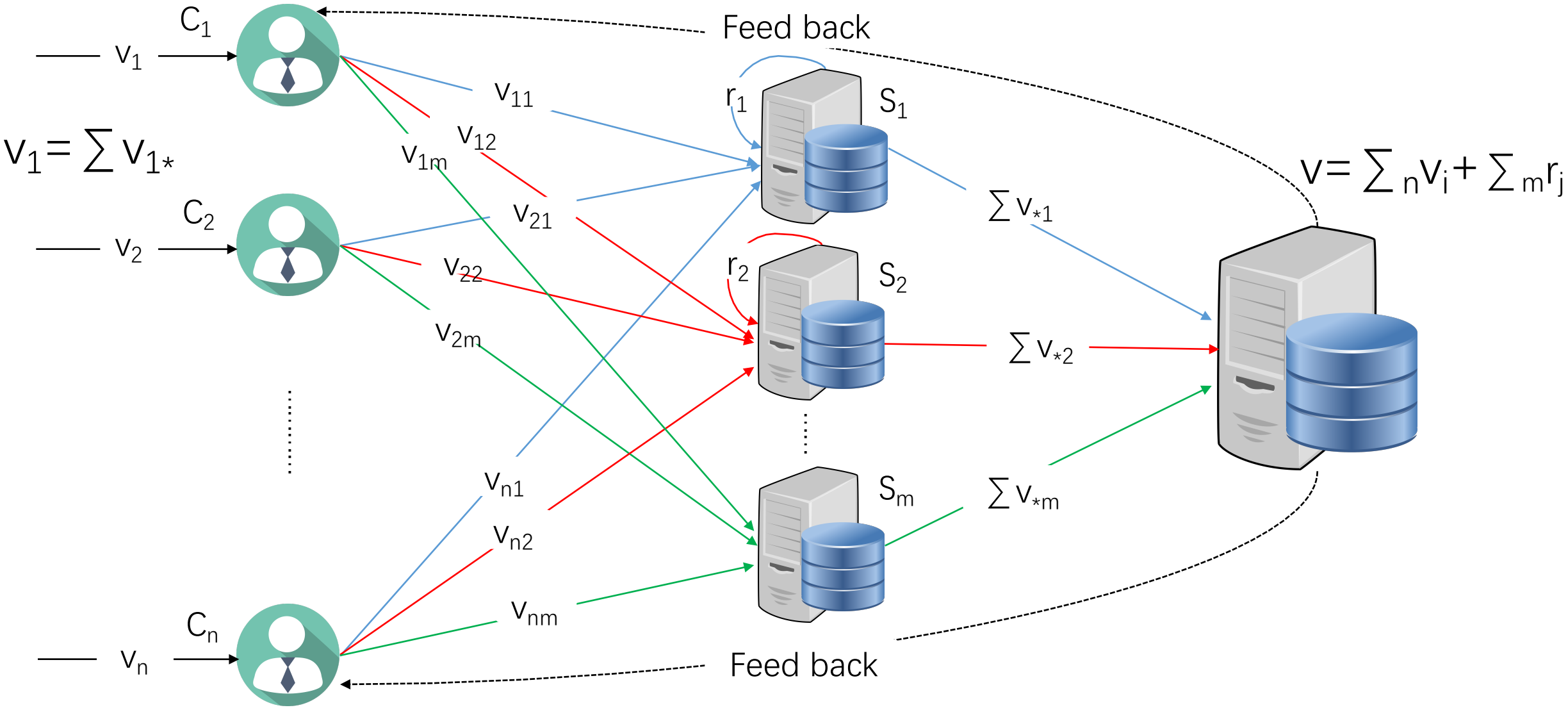
PEM: Noise-Free + Noise-based



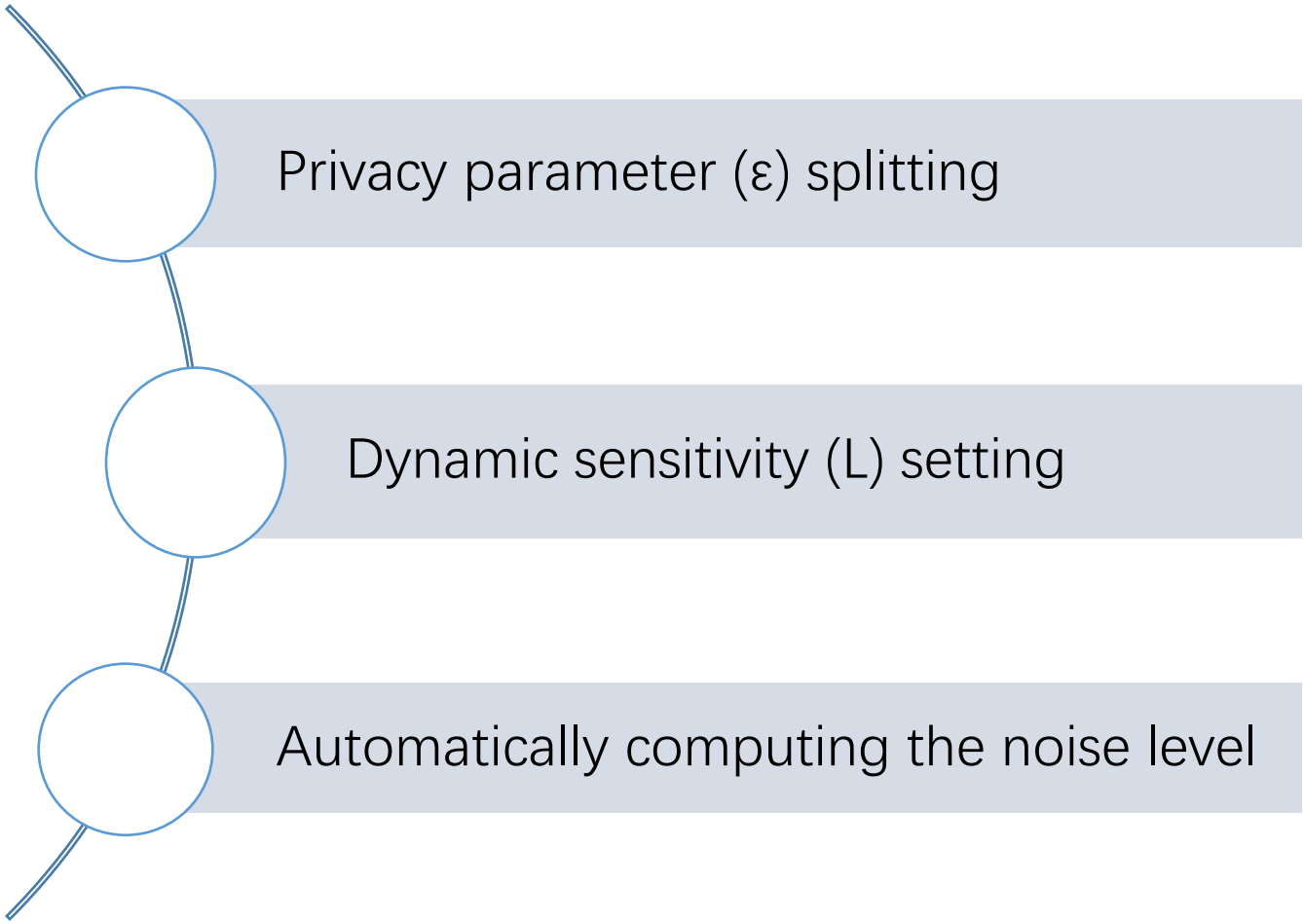
PEM: Noise-Free + Noise-based



Architecture Overview —— PEM



Utilize Algorithm Properties



Privacy parameter (ε) splitting

different privacy bonuses on different dimensions
e.g. k-means

Dynamic sensitivity (L) setting

sensitivity: the maximum distance of outputs of an algorithm on different inputs

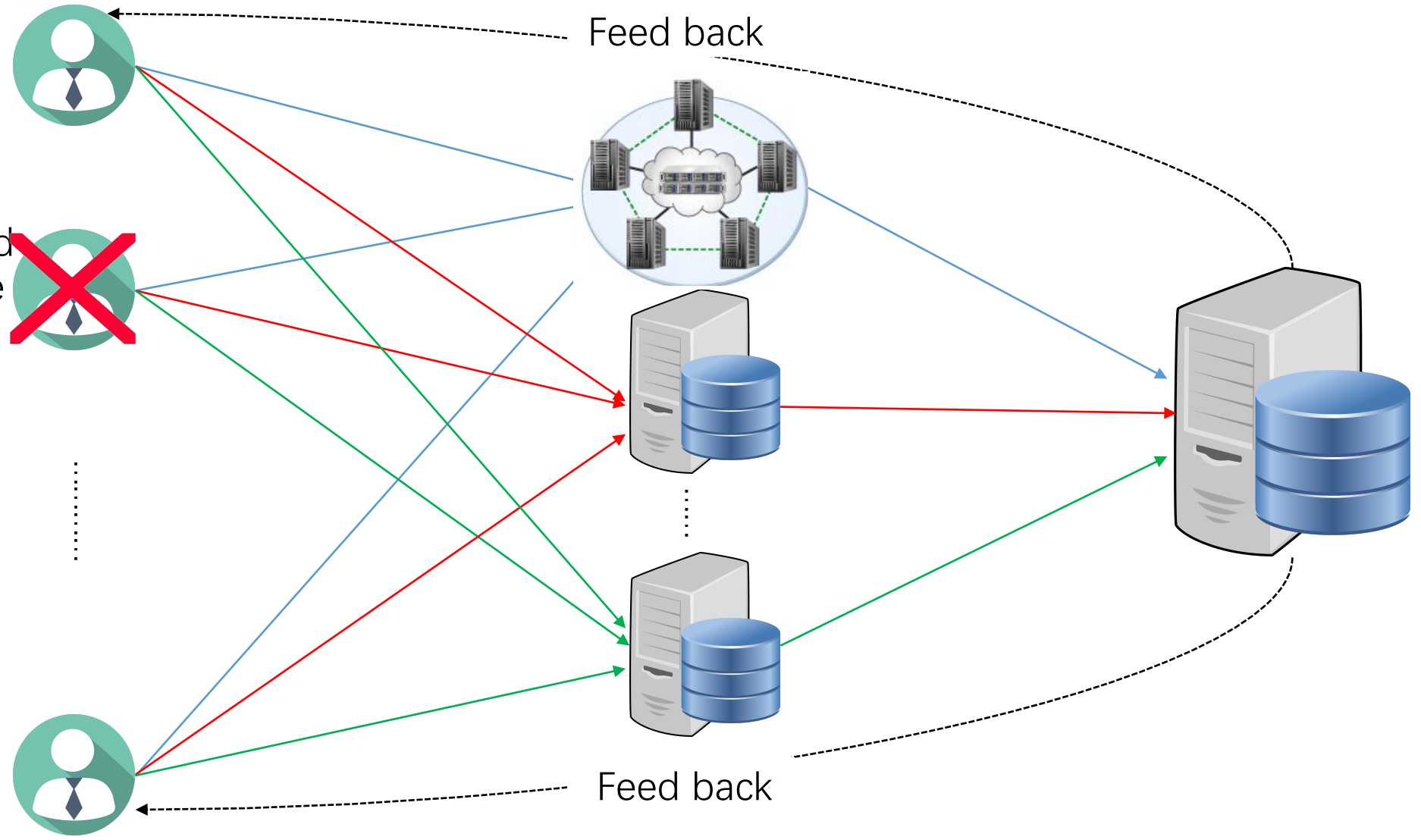
Sensitivity may change over iterations
e.g. Apriori

Automatically computing the noise level

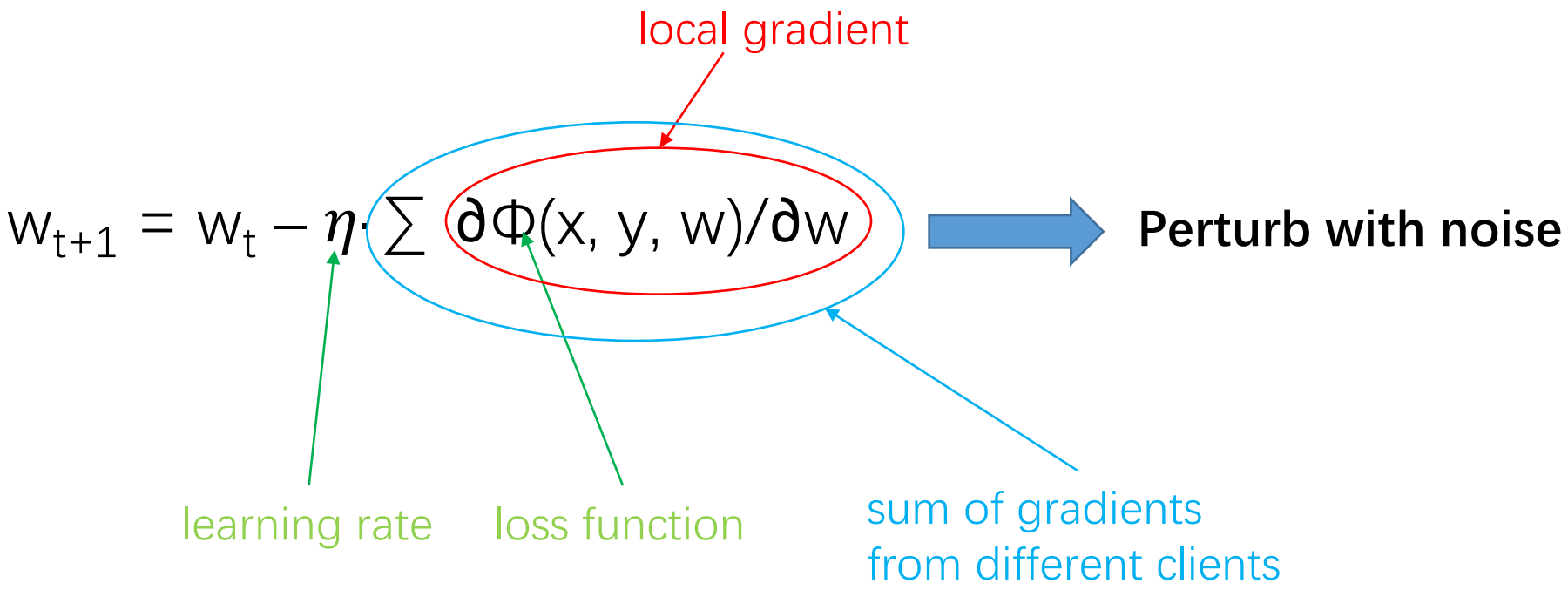
- ε-splitting: PEM generates different noise for different dimensions
- dynamic L setting: PEM computes λ(t) and generates noise r(t) = Lap(λ(t))

System Implementation

- Extend servers to clusters
- Handle client failures
- Intuitive configurations and programing interfaces (see below)
- Online processing
- Algorithm specific optimizations
  - sampling
  - sparse vector technique



Example Algorithms —— Gradient Descent



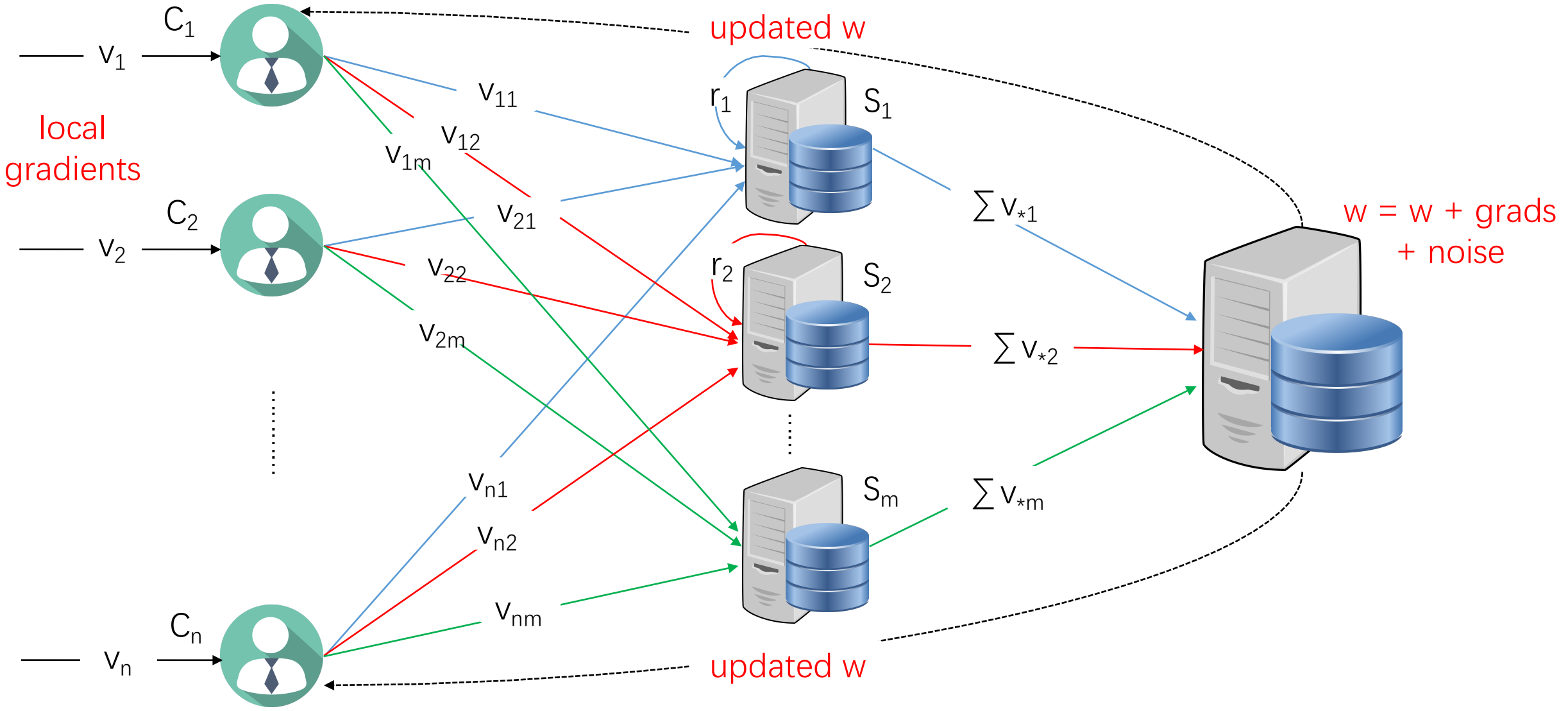
Example Algorithms —— Gradient Descent

Function *Init()*:
Initialize ϵ and L as constants. Initialize the iteration count T, the weight $\mathbf{w}$, loss function ϕ and the learning rate η.

Function $mapInClient(\mathbf{x}, y, \mathbf{w}, \phi)$:
return $\partial\phi(\mathbf{x}, y, \mathbf{w})/\partial\mathbf{w}$.

Function $reduceInAggregator(\mathbf{v}, \mathbf{x}, \eta, t)$:
update $\mathbf{w}$ as $\mathbf{w} = \mathbf{w} - \eta\frac{1}{\sqrt{t}} \cdot \frac{1}{\sum_{i=1}^{n} b_i}\mathbf{v}$.

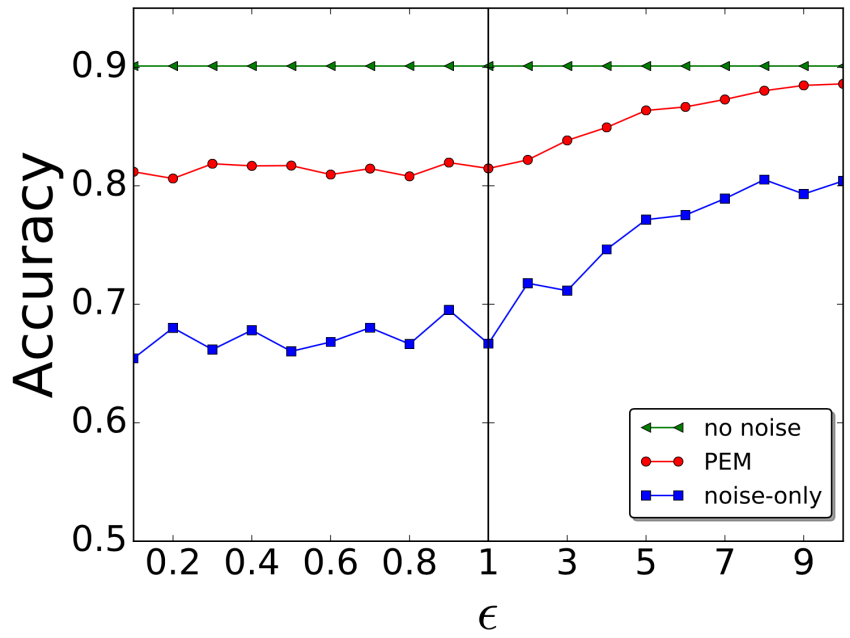
Example Algorithms —— Gradient Descent



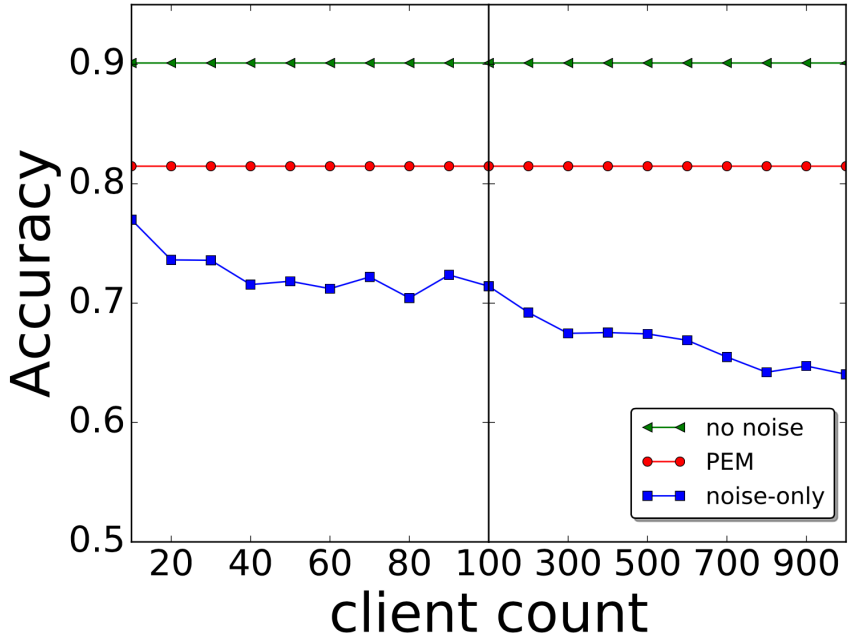
Example Algorithms —— Gradient Descent

- Dataset: Adult, 48842 instances, 124 dimensions
- Sensitivity: 28
- Training: 1000 iterations, 10-fold validation

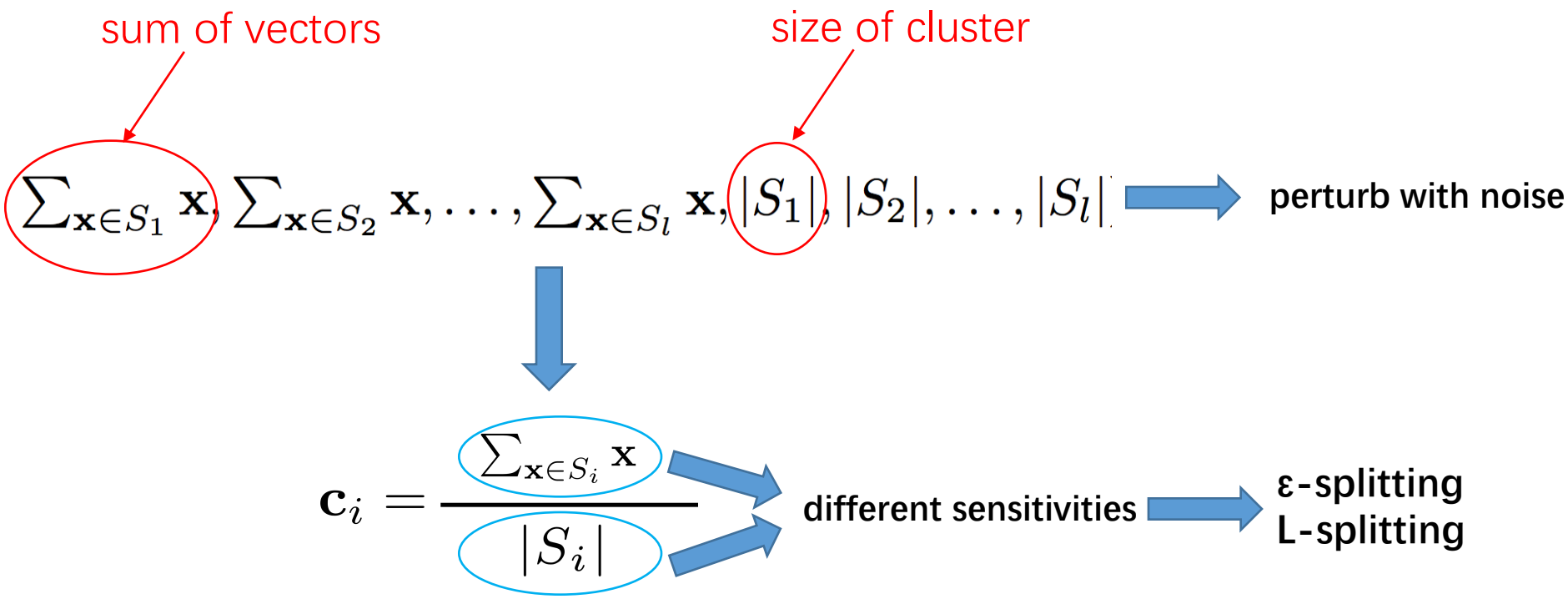
client count = 100



ε = 1



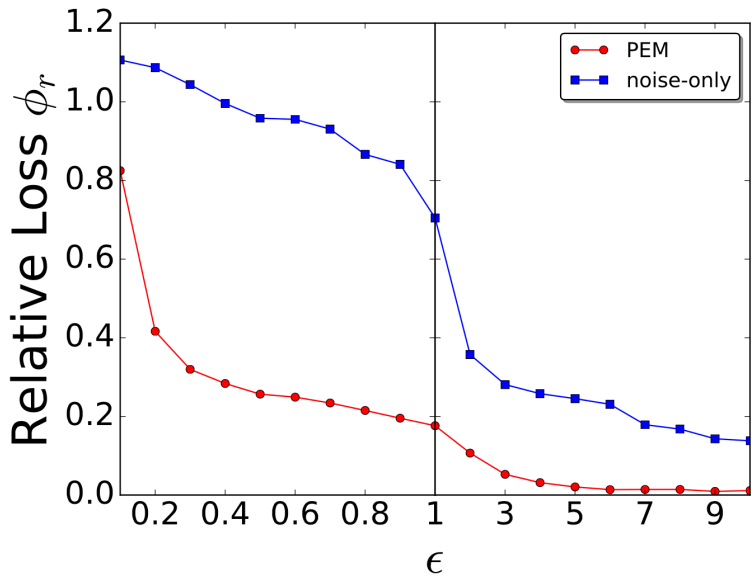
Example Algorithms —— k-means



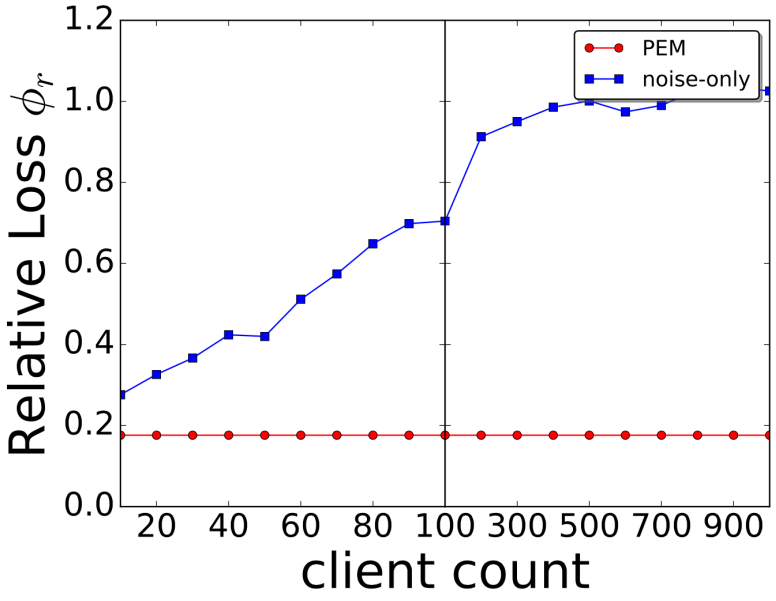
Example Algorithms —— k-means

- Dataset: Nursery, 21960 instances, 8 categorical attributes
- Sensitivity: $L_1 = 16$
- Training: 5 clusters, 50 iterations

client count = 100



ε = 1

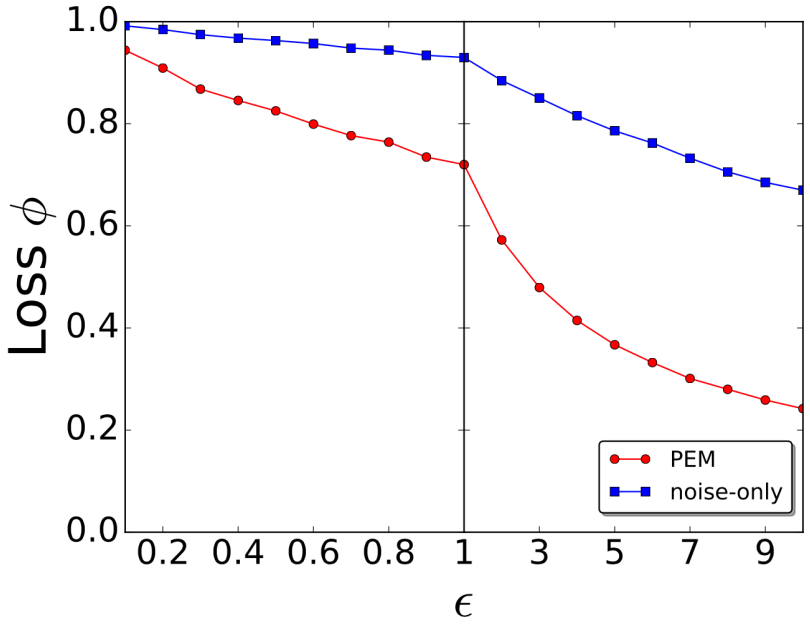


Example Algorithms —— Apriori

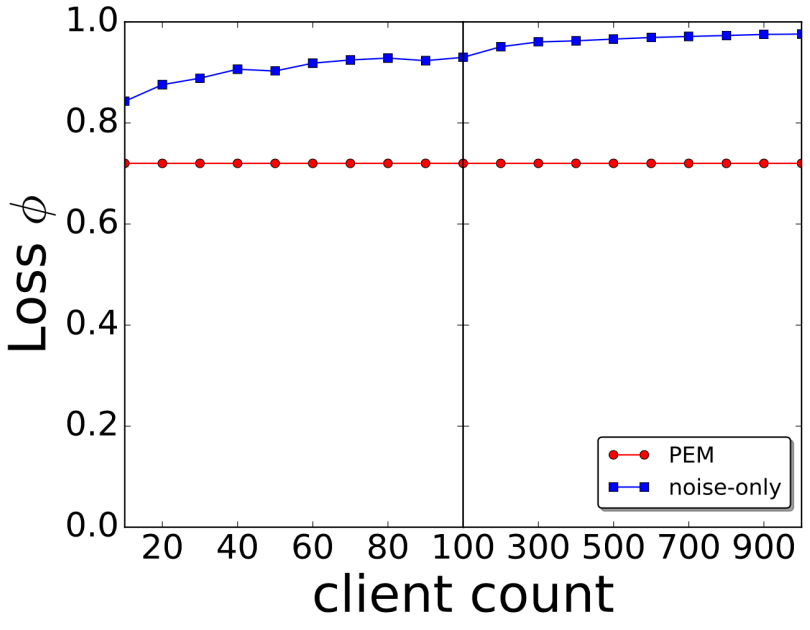
itemsets of different lengths have different sensitivities ➡ **dynamic L setting**

- Dataset: Mushroom, 8124 instances, 22 attributes
- Minimum support: 0.01
- Length of large itemsets: ⩽ 4
- Loss function: $\phi(\mathbf{I}) = \frac{\sum_k \sum_{t \in I_k \cup I_{k0}} |I_k(t) - I_{k0}(t)|}{\sum_k \sum_{t \in I_k \cup I_{0k}} |I_k(t) + I_{k0}(t)|}$

client count = 100

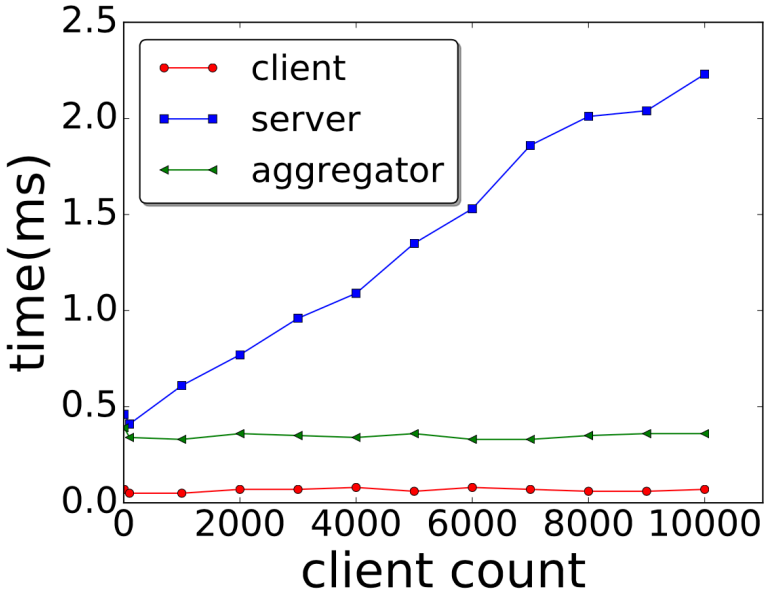


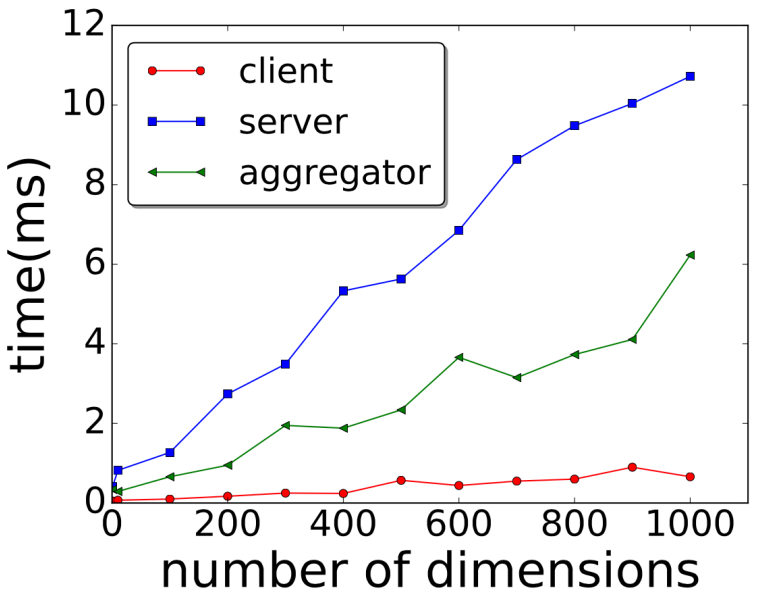
ε = 1



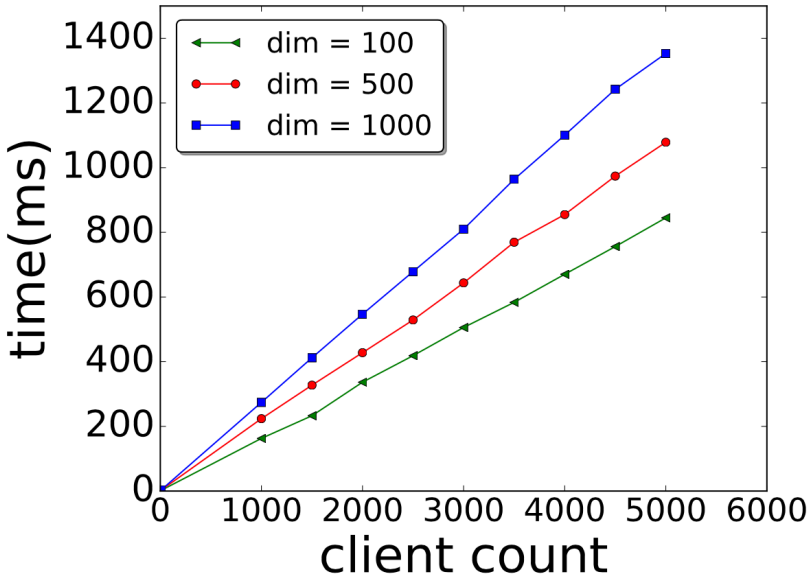
High Efficiency and Scalability

computation cost





computation + communication



Conclusion and Future Work

- PEM: a tradeoff among privacy, utility and computation overhead
  - **simple assumptions**: only a few semi-honest servers
  - supports a large range of **common applications**
  - entails **low computation overhead**
  - **intuitive configurations**

- **Good accuracy**: PEM achieves the same level of privacy without the amount of accuracy degradation that previous systems suffer from

- Future Work
  - support for more operations, e.g. handling vertically partitioned datasets
  - permission system allowing different clients to see different levels of private content