Privacy-preserving Data Mining and Machine Learning

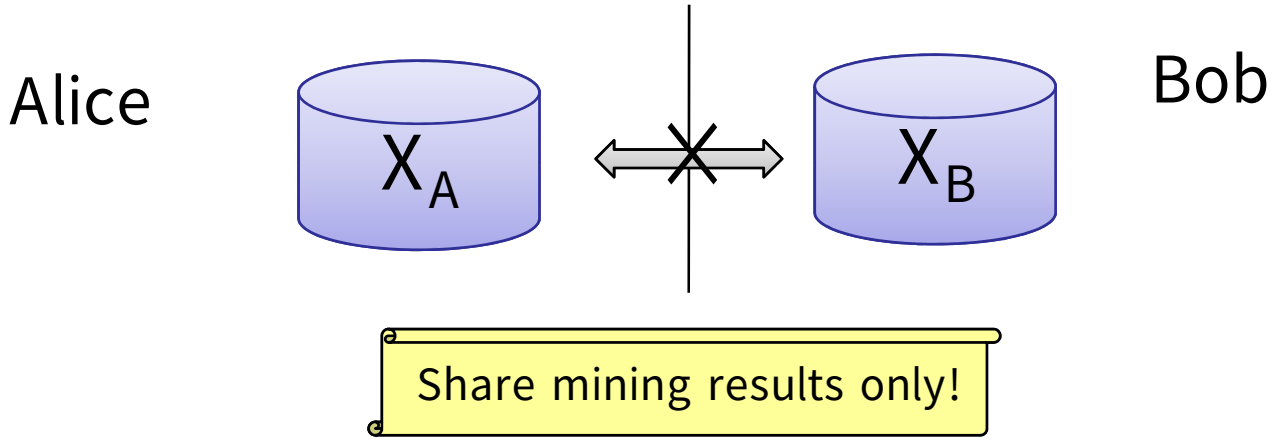
Tokyo Institute of Technology, Jun Sakuma

Agenda

- **Basic concepts of PPDM**
  - Secure multiparty computation, trusted third party
  - Randomized approach, cryptographic approach
- **PPDM: Cryptographic approach**
  - Definition
  - Proof methodology
  - Building blocks
- **Studies from our group**
  - Privacy-preserving k-means clustering in P2P (PAKDD2008)
  - Privacy-preserving reinforcement learning (ICML2008)
- **Future direction of PPDM/ML**

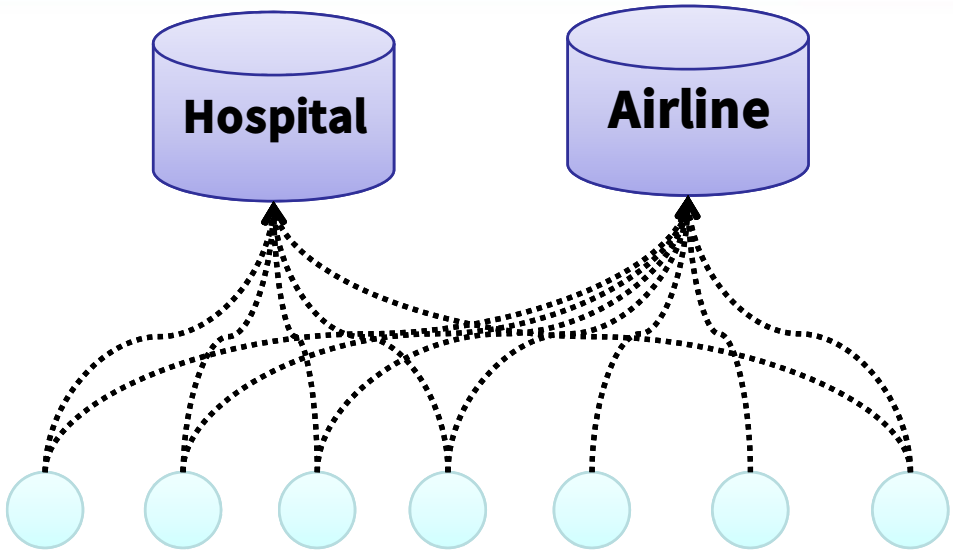
Basics of PPDM

Privacy-preserving Data Mining



- **Alice holds a private dataset X_A**
- **Bob holds a private dataset X_B**
- **Privacy-preserving data mining (PPDM) problem:**
  - Both do not wish to share their datasets
  - They wish to execute a specific data mining over joint dataset $X_A \cup X_B$
  - At the end, they wish to share only mining results

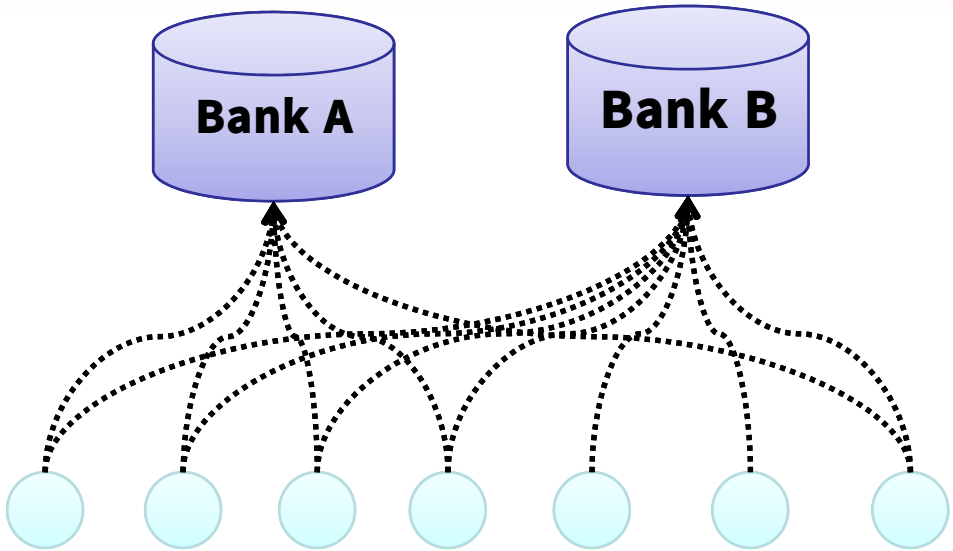
Scenario 1: Identification of epidemic source



individuals

- Outbreak of epidemic
  - Hospitals wish to identify the source of epidemic geographically
  - Clustering of patients' medical records with geographical movement
- Integration of databases is difficult due to…
  - privacy preservation
  - legal constraints
- How can we do clustering without sharing private information?

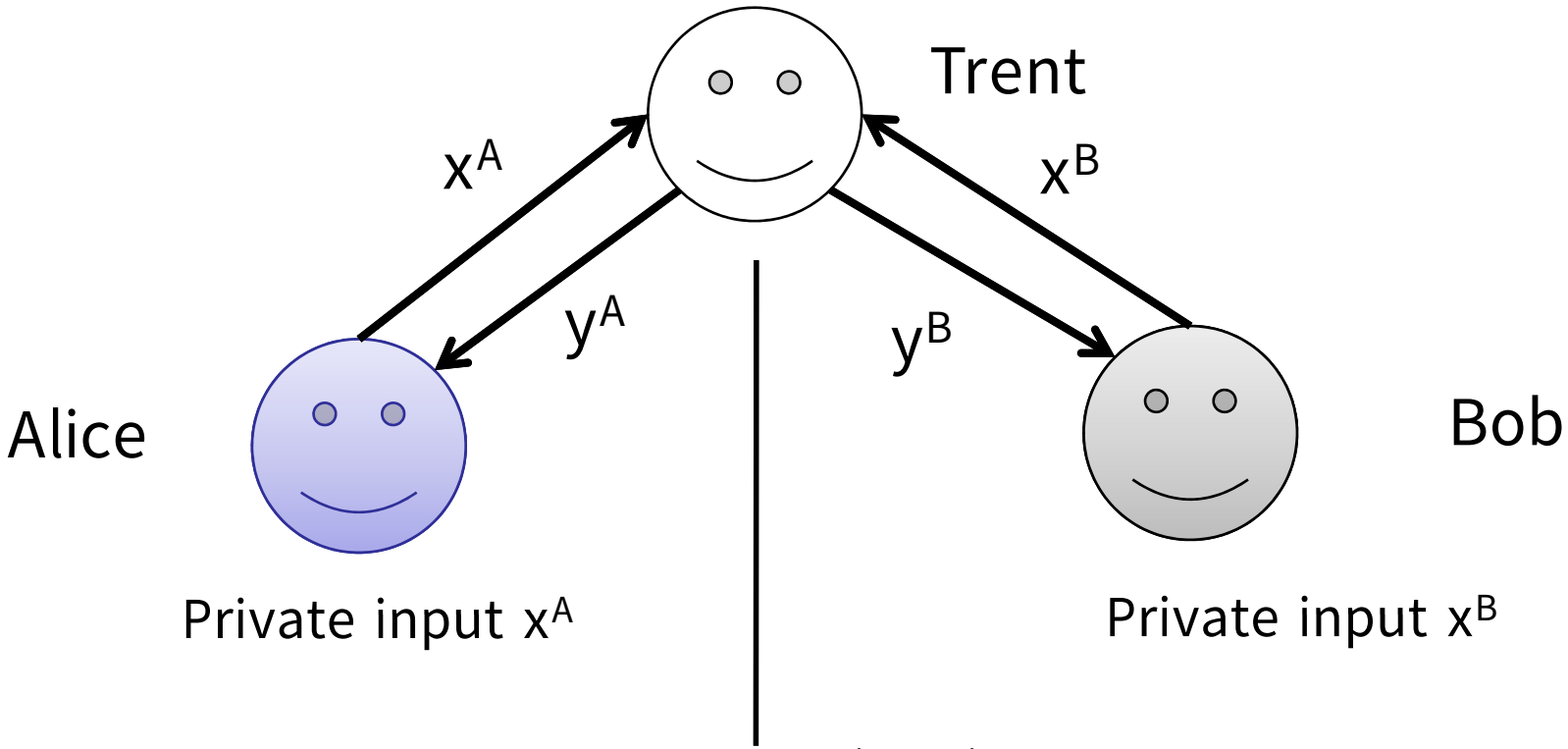
Scenario 2: Bankruptcy Prediction



enterprises

- Bankruptcy prediction
  - Enterprises have accounts of banks
  - Banks wish to predict the probability of bankruptcy with joint transactions
- Integration of transactions is difficult due to…
  - confidentiality
  - legal constraints
- How can banks predict the bankruptcy without sharing confidential information?

Trusted Third Party



- **Introduce a trusted third party (TTP): Trent**
- **Trent processes any specified computation**
- **Trent is always faithful to the specified protocol**

Trusted Third Party

- **TTP is good, but need to facilitate an authority**
- **Question: Can we do computation in a standard setting?**
  - No authority
  - Regular network (e.g., TCP/IP)

Secure Multiparty Computation

- **Yao's secure two-party computation [Yao86]**
  - Assumption: parties are semi-honest
    - Do not deviate form the protocol
    - Attempt to learn extra information from the message transcripts
  - Yao's protocol
    - ***Any* computation** can be made private
    - Complexity is polynomially bounded by the size of the circuit that evaluates the specified function

Secure Multiparty Computation

- **Is Yao the final answer?**
  - Yao is good but too costly, particularly when
    - Large input, (x^A, x^B)
    - Large circuit (complex computation)

	Elapsed exec time (sec) in LAN	Elapsed exec time (sec) in WAN
Bit-and operation	0.41	2.57
Billionaires (comparison of two 32bit integers)	1.25	4.01
Key database search (search an item from 16 items with 6bit key)	0.49	3.38
Median (find median form 10 16bit integers)	7.09	16.63

Malkhi, D. et al, Fairplay - a secure two-party computation system, Proc. of the 13th USENIX Security Symposium, 287– 302, 2004

Two approaches for PPDM

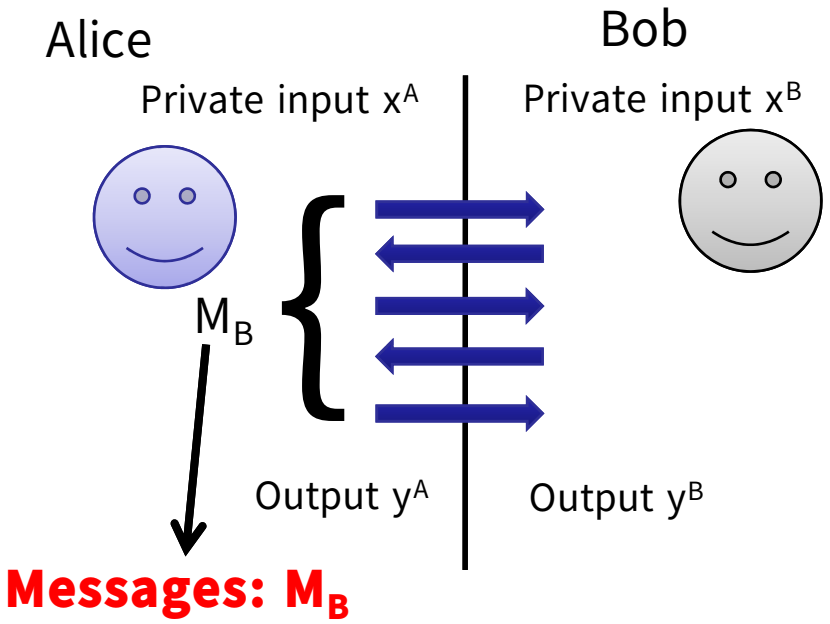
- **Randomization approach** [**Agrawal et al. *SIGMOD2000***]
  - Add random perturbation to original data
  - Apply data mining algorithm
  - Remove the perturbation effect (maximum likelihood)
- **Cryptographic approach** [**Lindel et al. *CRYPTO2000***]
  - Composition of security protocols, including Yao
    - Mostly Yao is used for a small portion of the entire computation

Comparison of approaches

	accuracy	comp. cost	authority	generality	privacy
TTP	perfect	small	required	general	provably secure
SMC (Yao)	perfect	large	not required	general	provably secure
Random	approx.	small	not required	limited	statistically secure
Crypto	perfect	medium	not required	limited	provably secure

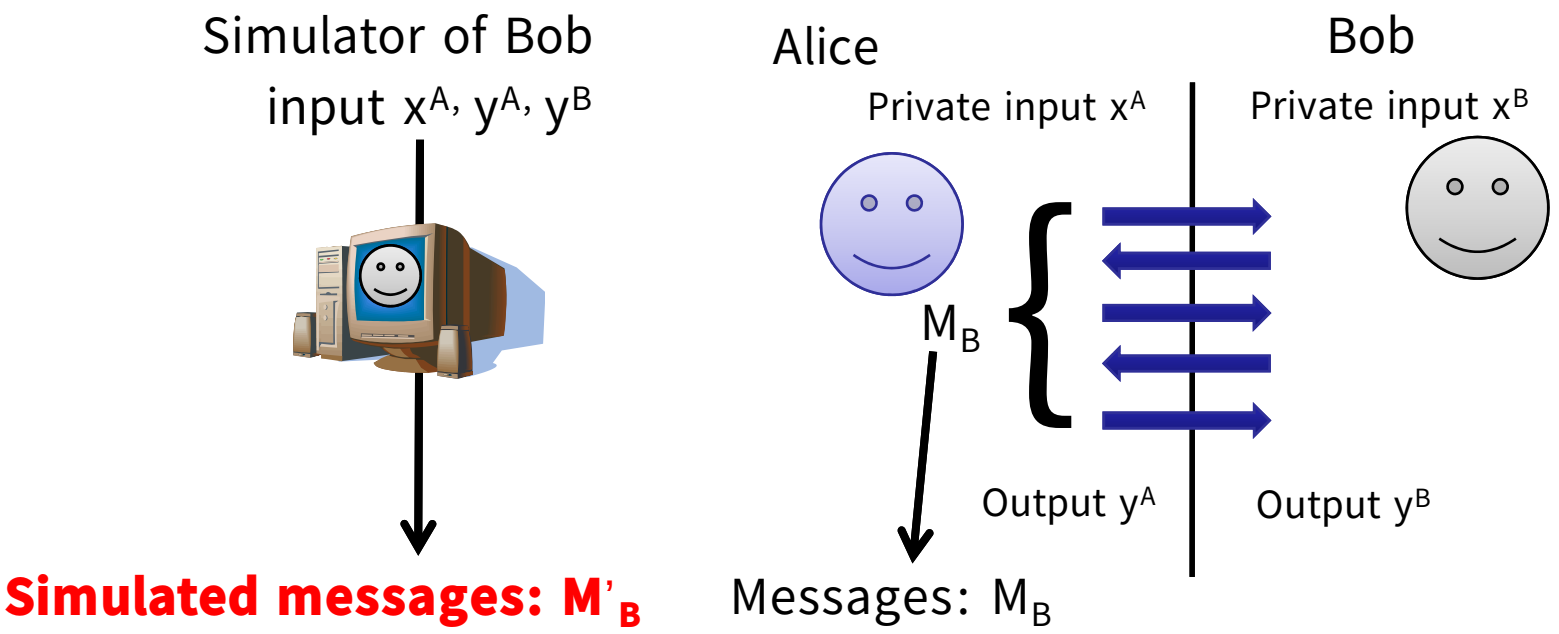
Cryptographic approach

Privacy-preservation in distributed computation



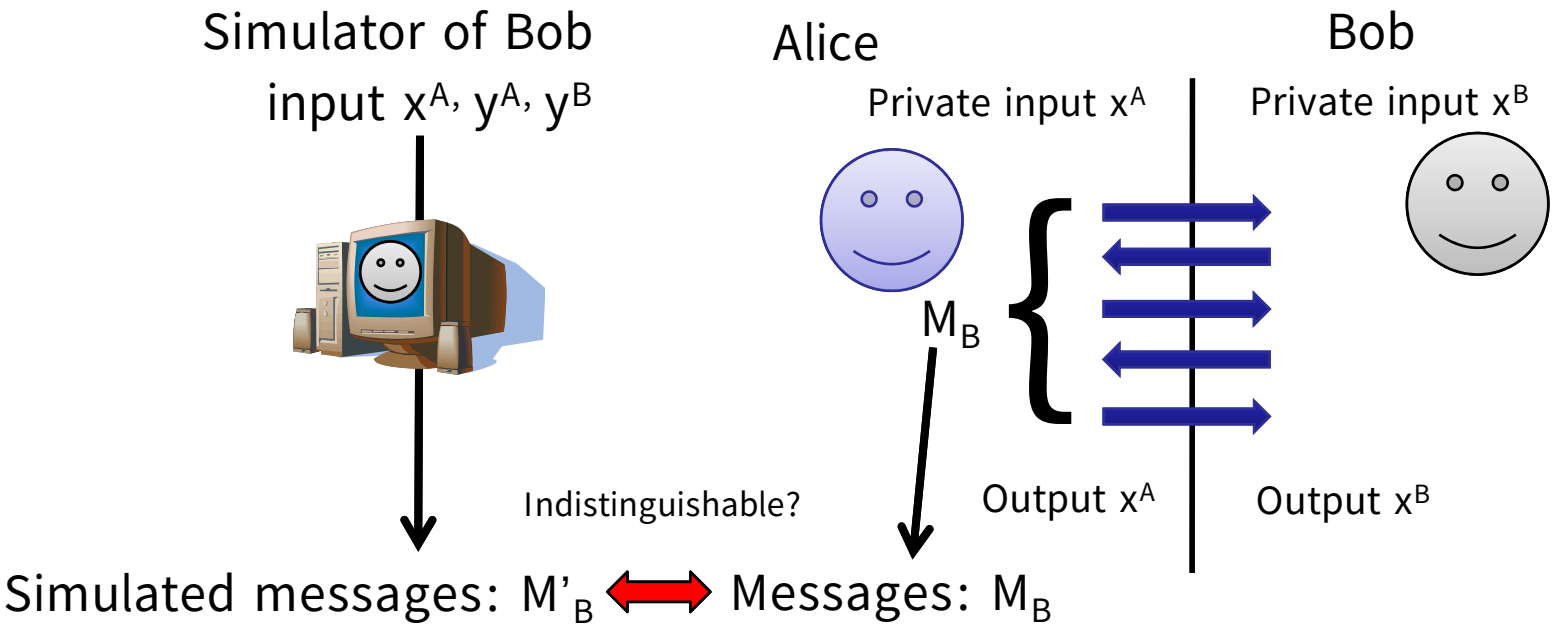
- M_B : all message transcripts Alice received

Privacy-preservation in distributed computation



- M_B : all message transcripts Alice received
- M_B': messages generated by simulator S given only Alice's input and the protocol output

Privacy-preservation in distributed computation

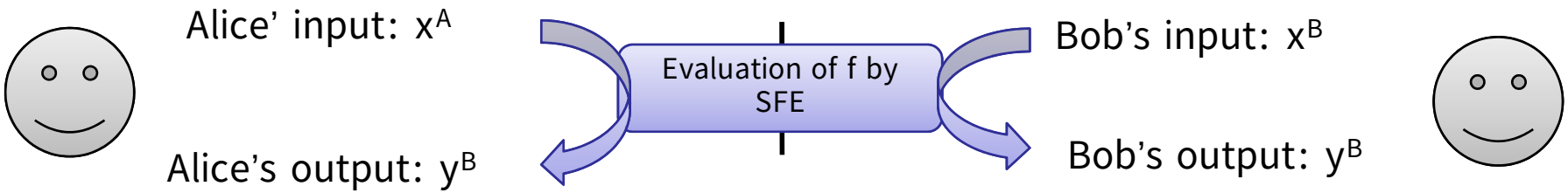


- M_B : all message transcripts Alice received
- M_B': messages generated by simulator S given only Alice's input and the protocol output
- Privacy-preservation
  - The protocol does not reveal unnecessary information of Bob if M_B and M_B' are indistinguishable

Security protocols as building blocks

- Secret Sharing
- Oblivious Transfer
- Secure function evaluation (=Yao's protocol)
- Homomorphic public-key cryptosystem
- Oblivious polynomial evaluation
- Secure set intersection, etc…

Secure Function Evaluation



- **Secure function evaluation (including Yao's protocol)**
  - Function: $f : X \times X \mapsto Y \times Y$
  - For any f, SFE enables private evaluation of $(y^A, y^B) \leftarrow f(x^A, x^B)$
- **How does SFE works? (intuitively)**
  - Convert function f to a circuit
  - Use secure computation of "and" and "or" (garbled circuit)
- **Example**
  - Private comparison: Which is greater x^A or x^B?
  - Private matching: Is element x^A included in list x^B?

Homomorphic Public-key Cryptosystem

- Let m ϵ Z_N be a message and r ϵ Z_N be a random number
- Let (pk, sk) be a pair of public and secret key. Then,
  - Encryption: $c \leftarrow \mathrm{Enc}_{p_k}(m_0; r_0)$
  - Decryption: $m_0 \leftarrow \mathrm{Dec}_{s_k}(c)$
- Let m0,m1, r1,r2 ϵ Z_N
- Homomorphic cryptosystem allows:
  - Addition of encrypted values

$$\mathrm{Enc}_{p_k}(m_0; r_0) \cdot \mathrm{Enc}_{p_k}(m_1; r_1) = \mathrm{Enc}_{p_k}(m_0 + m_1; r_1 \cdot r_2)$$

  - Multiplication of encrypted value and a plain value

$$\mathrm{Enc}_{p_k}(m_0; r_0)^{m_1} = \mathrm{Enc}_{p_k}(m_0 m_1; r')$$

e.g. Paillier cryptosystem[Pai00]

Example: private computation of ax+y

Alice has x **Bob has y, a**

Problem: compute random shares of ax+y

Example: private computation of ax+y

Alice has x **Bob has y, a**

Problem: compute random shares of ax+y

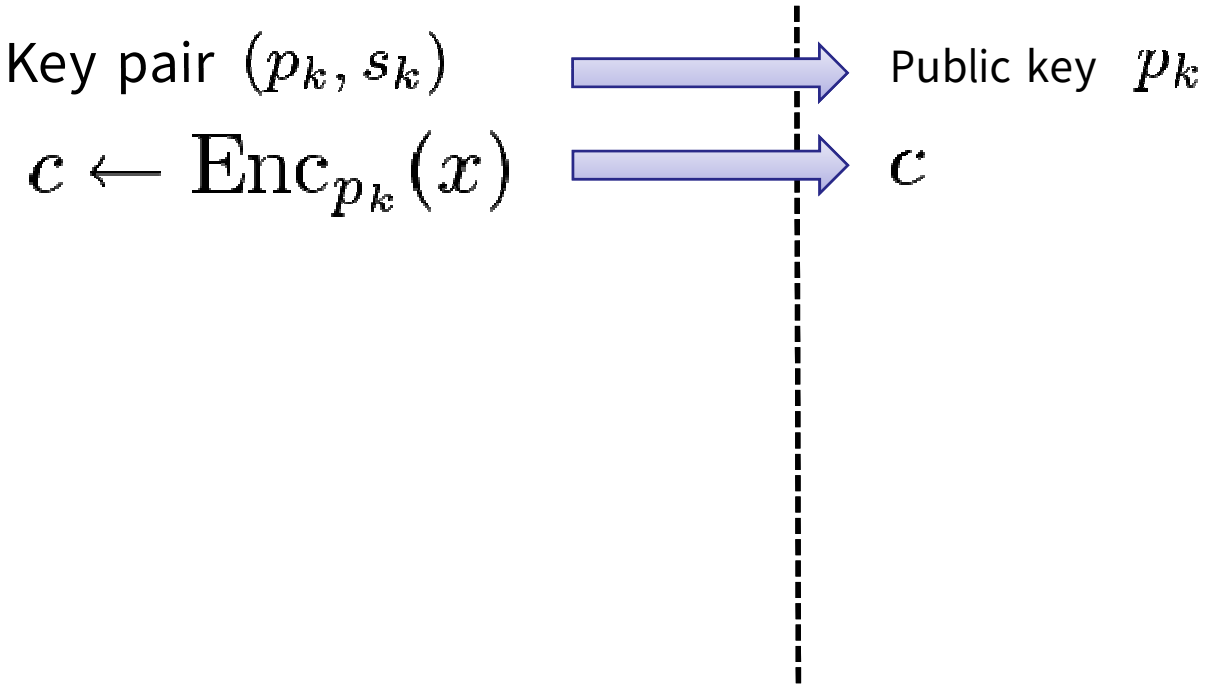
Key pair (p_k, s_k)

$$c \leftarrow \mathrm{Enc}_{p_k}(x)$$

Example: private computation of ax+y

Alice has x **Bob has y, a**

Problem: compute random shares of ax+y



Example: private computation of ax+y

Alice has x **Bob has y, a**

Problem: compute random shares of ax+y

Key pair (p_k, s_k)

$c \leftarrow \mathrm{Enc}_{p_k}(x)$

Public key p_k

c

Generate a random number r_B

Example: private computation of ax+y

Alice has x **Bob has y, a**

Problem: compute random shares of ax+y

Key pair (p_k, s_k)

$c \leftarrow \mathrm{Enc}_{p_k}(x)$

Public key p_k

c

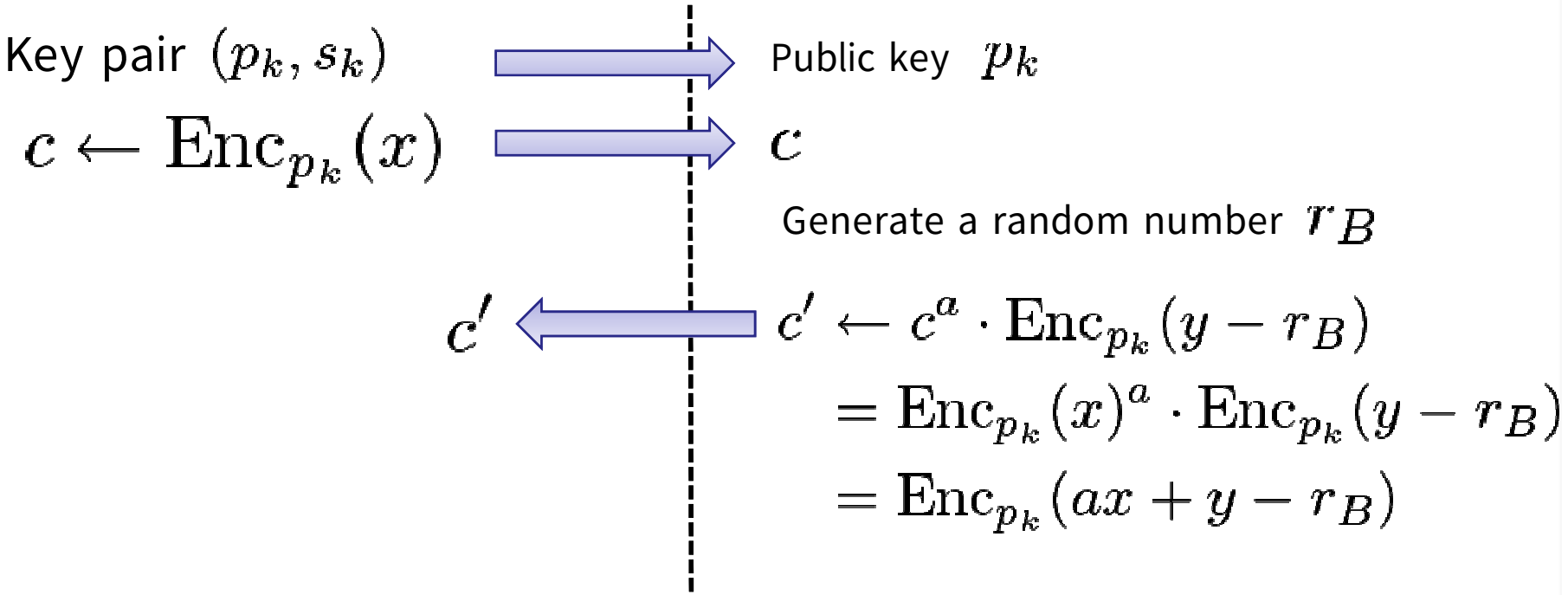
Generate a random number r_B

$$c' \leftarrow c^a \cdot \mathrm{Enc}_{p_k}(y - r_B)$$
$$= \mathrm{Enc}_{p_k}(x)^a \cdot \mathrm{Enc}_{p_k}(y - r_B)$$
$$= \mathrm{Enc}_{p_k}(ax + y - r_B)$$

Example: private computation of ax+y

Alice has x **Bob has y, a**

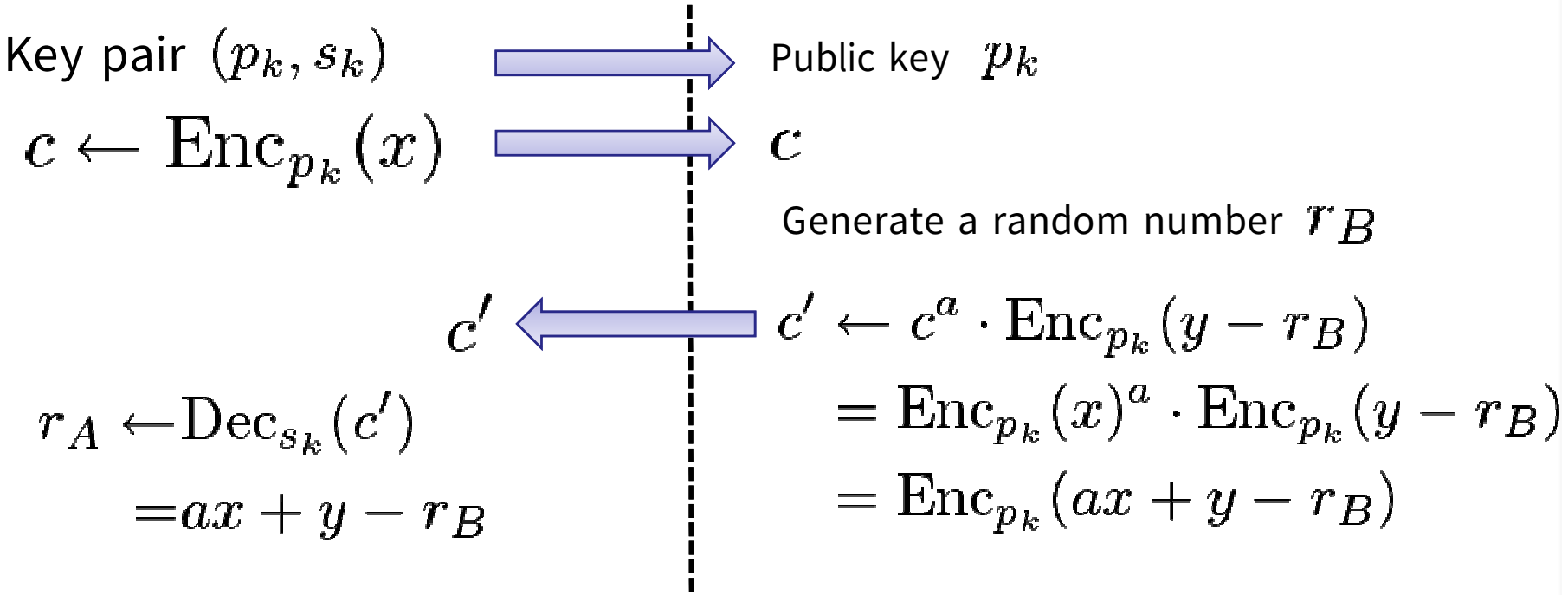
Problem: compute random shares of ax+y



Example: private computation of ax+y

Alice has x **Bob has y, a**

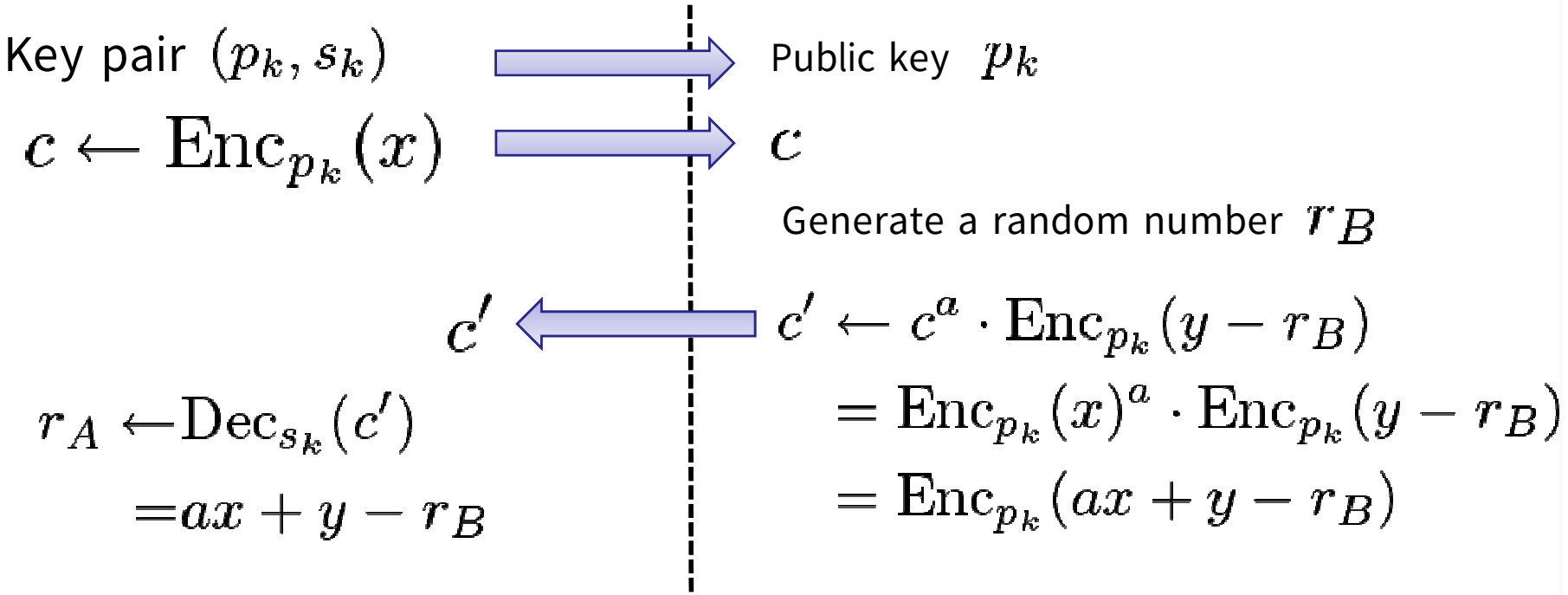
Problem: compute random shares of ax+y



Example: private computation of ax+y

Alice has x **Bob has y, a**

Problem: compute random shares of ax+y



$$r_A + r_B = ax + y \mod N$$

Example: private comparison of scalar product

Alice $\boldsymbol{y} = (y_1, \ldots, y_q)$ **Bob** $\boldsymbol{x}^1 = (x_1^1, \ldots, x_q^1)$ $\boldsymbol{x}^2 = (x_1^2, \ldots, x_q^2)$

Problem: which is greater $\boldsymbol{x}^1 \cdot y$ or $\boldsymbol{x}^2 \cdot y$?

Example: private comparison of scalar product

Alice $\boldsymbol{y} = (y_1, \ldots, y_q)$ **Bob** $\boldsymbol{x}^1 = (x_1^1, \ldots, x_q^1)$, $\boldsymbol{x}^2 = (x_1^2, \ldots, x_q^2)$

Problem: which is greater $\boldsymbol{x}^1 \cdot y$ or $\boldsymbol{x}^2 \cdot y$?

Key pair (p_k, s_k) → Public key p_k

$(\mathrm{Enc}_{p_k}(y_1), \ldots, \mathrm{Enc}_{p_k}(y_q))$ → $(c_1, \ldots, c_q)$

Generate a random number r^B

$$w \leftarrow \mathrm{Enc}_{p_k}(-r^B) \cdot \prod_{i=1}^{q} c_i^{x_i^1 - x_i^2}$$

$$= \prod_{i=1}^{q} \mathrm{Enc}_{p_k}(y_i(x_i^1 - x_i^2) - r^B)$$

$$= \mathrm{Enc}_{p_k}(\boldsymbol{x}^1 \cdot \boldsymbol{y} - \boldsymbol{x}^2 \cdot \boldsymbol{y} - r^B)$$

$$r^A \leftarrow \mathrm{dec}_{s_k}(w)$$

$$= \boldsymbol{x}^1 \cdot \boldsymbol{y} - \boldsymbol{x}^2 \cdot \boldsymbol{y} - r^B$$

r_A and r_B are random shares of $\boldsymbol{x}^1 \cdot \boldsymbol{y} - \boldsymbol{x}^2 \cdot \boldsymbol{y}$

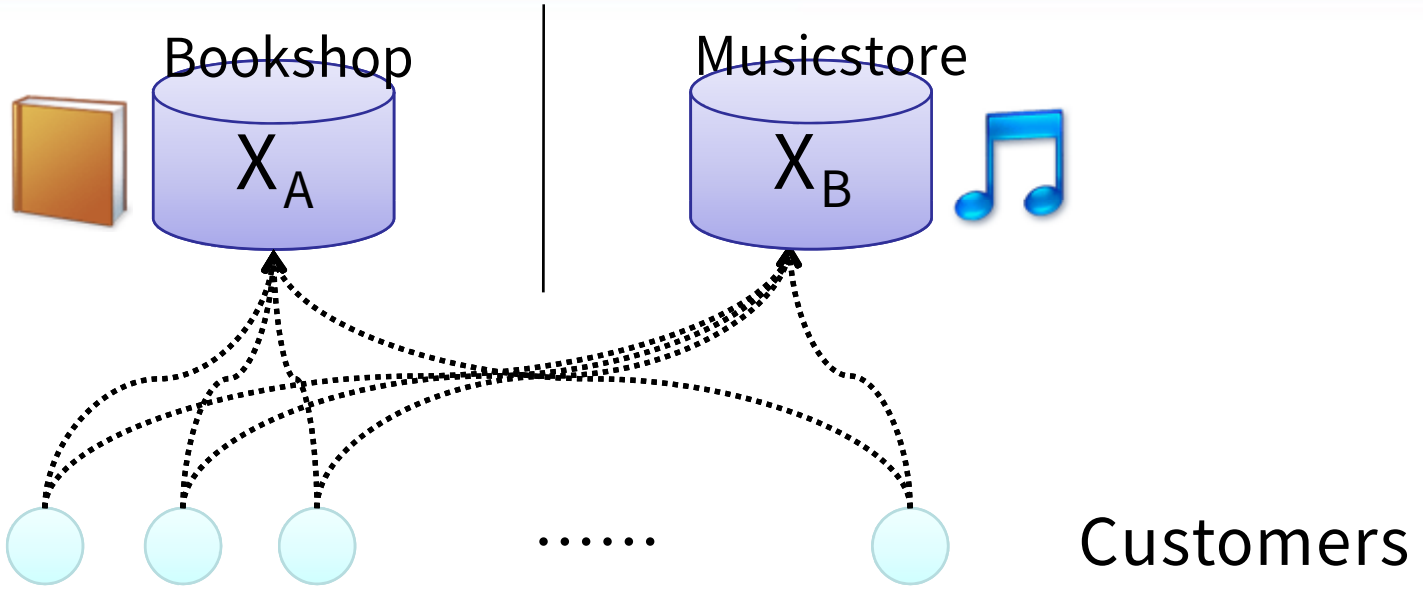
r_A → Comparison by SFE ← r_B

Comparison by SFE → void

Comparison by SFE → Comparison result

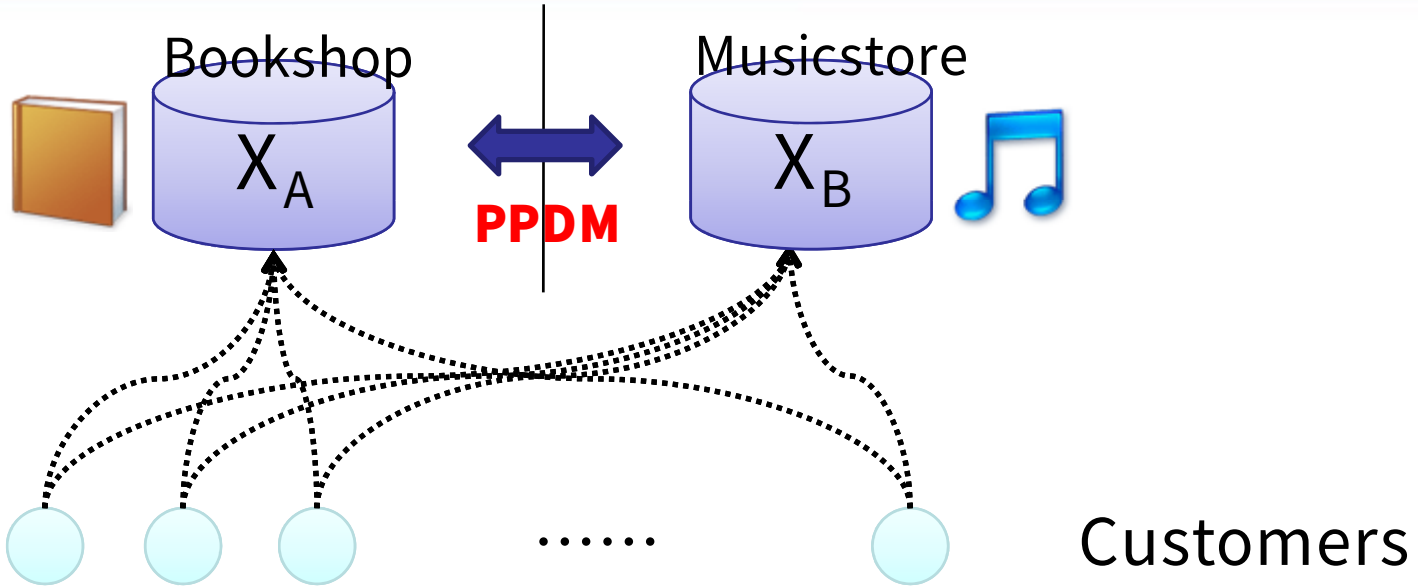
PPDM for k-means

Book shop & music store example



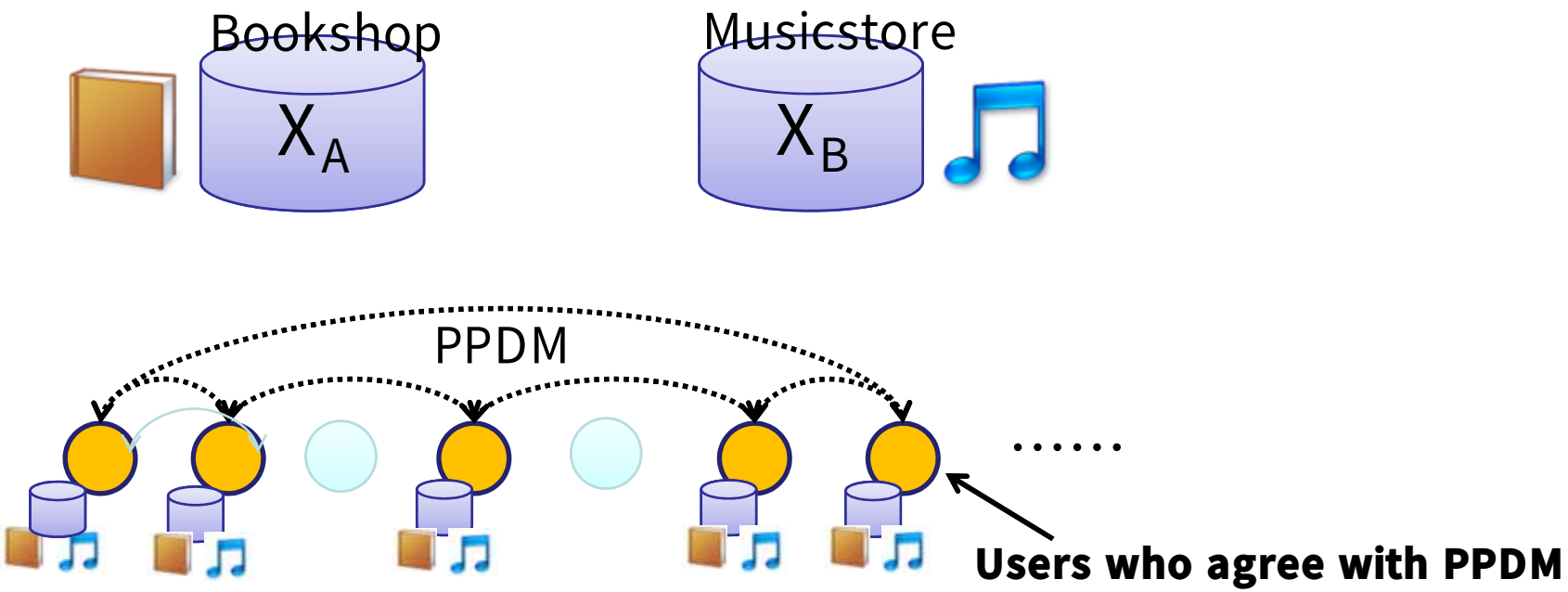
- Users deposit their personal data to organizations
- Based on customers' personal records:
  - Bookshop offers us a recommendation "Customers buy this book with…"
  - Music store offers us a recommendation "Customers buy this music with…"
- **Can we know "Customers who buy book A loves music B" without sharing our private information?**

Book shop & music store example



- Server-centric privacy-preservation
  - If the book shop and the music store reach agreement of PPDM, it would be possible
  - If not, collaboration between the book shop and the music store may not happen
- We cannot exploit our private information by ourselves
- How can we privately manage our data by ourselves?

User-centric Privacy Preservation

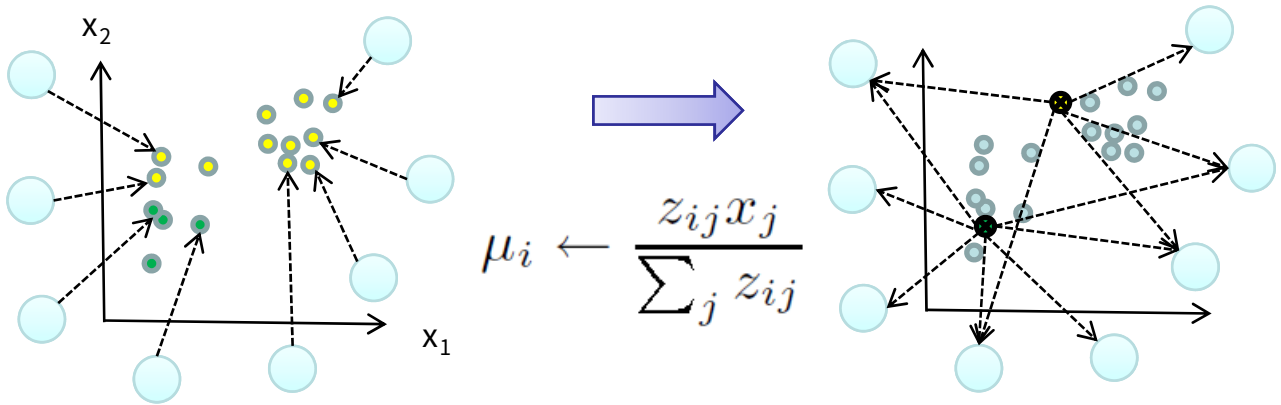


- User-Centric Privacy-preservation
  - Run the protocol among users who agree with PPDM
- The challenge
  - Scalability: the number of parties would be hundreds or more
- Our solution: peer-to-peer network

P2P K-means clustering

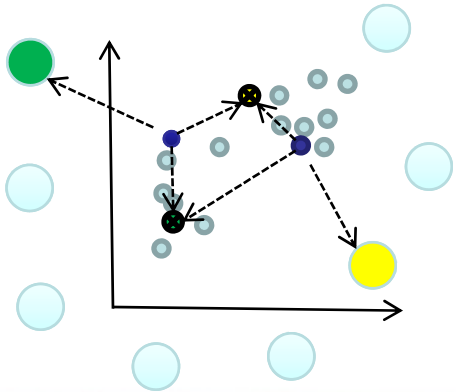
- **alternate iterations of…**

1. Update cluster center to the mean vector (global)



$$\mu_i \leftarrow \frac{z_{ij}x_j}{\sum_j z_{ij}}$$

2. Update cluster label to the nearest cluster center (local)



$$z_{ij} \leftarrow \begin{cases} 1 & \text{If } i = \arg\min_\ell d(x_j, \mu_\ell) \\ 0 & \text{o.w.} \end{cases}$$

Problem statement

- **Node P_j has a datapoint x_j and cluster label z_{ij}**
- **Problem 1:**
  - Compute cluster center without sharing x_i and z_{ij} among nodes

$$\mu_i \leftarrow \frac{z_{ij} x_j}{\sum_j z_{ij}}$$

  - Use gossip-based aggregation and homomorphic cryptography
- **Problem 2:**
  - Compute cluster label without sharing x_j and z_{ij} among nodes

$$z_{ij} \leftarrow \begin{cases} 1 & \text{If } i = \arg\min_\ell d(x_j, \mu_\ell) \\ 0 & \text{o.w.} \end{cases}$$

  - Use Yao (private comparison) and homomorphic cryptography

Gossip-based aggregation[Kowalczyk05]

- Asynchronous averaging without central server
- Node P_j owns data x_j
  - Initialization

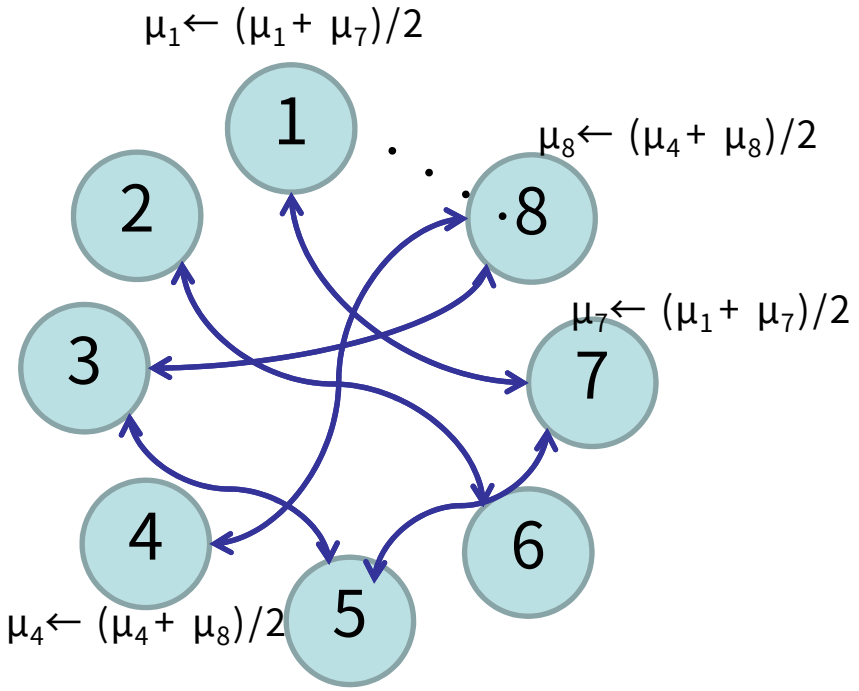
    $$\mu_j \leftarrow x_j$$

  - Contact with a node chosen uniform randomly;
  - Update the local estimate as the average of two estimates

    $$\mu_j \leftarrow \frac{\mu_j + \mu_{j'}}{2}$$

  - Convergence:

    $$\mu_j \rightarrow \mu \text{ as } t \rightarrow \infty$$

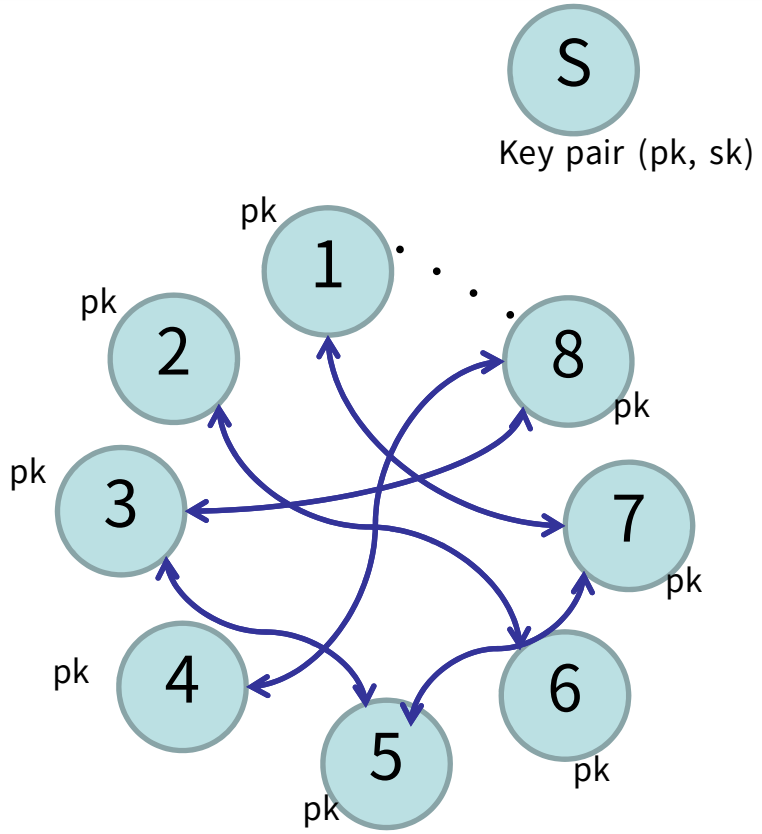


Privacy-preserving Gossip-based aggregation

- Server generates a key pair and distributes the public key
- Node P_j owns data x_j
  - Initialization

    $$c_j \leftarrow \mathrm{Enc}_{pk}(x_j)$$

  - Contact with a node chosen uniform randomly;



Cryptographic Gossip-based aggregation

- **Update in regular gossip based aggregation**

$$\mu_j \leftarrow \frac{\mu_j + \mu_{j'}}{2},$$

Cryptographic Gossip-based aggregation

- **Update in gossip-based aggregation**

$$\mu_j \leftarrow \frac{\mu_j + \mu_{j'}}{2},$$

- **Update without division**

$$\mu_j \leftarrow \mu_j + \mu_{j'}$$ (converges to $2^T \mu$)

Cryptographic Gossip-based aggregation

- **Update**

$$\mu_j \leftarrow \frac{\mu_j + \mu_{j'}}{2},$$

- **Asynchronous update without division**

$$\begin{aligned} \mu_j &\leftarrow \mu_j + 2^{t_j - t_{j'}} \mu_{j'}, \quad \text{if } t_j \geq t_{j'} \\ \mu_j &\leftarrow \mu_{j'} + 2^{t_{j'} - t_j} \mu_j, \quad \text{otherwise.} \end{aligned}$$

  - t_j : the number of messaging
  - Convergence to $2^T \mu$

Cryptographic Gossip-based aggregation

- **Update**

$$\mu_j \leftarrow \frac{\mu_j + \mu_{j'}}{2},$$

- **Asynchronous update without division**

$$\begin{aligned} \mu_j &\leftarrow \mu_j + 2^{t_j - t_{j'}}\,\mu_{j'}, \quad \text{if } t_j \geq t_{j'} \\ \mu_j &\leftarrow \mu_{j'} + 2^{t_{j'} - t_j}\,\mu_j, \quad \text{otherwise.} \end{aligned}$$

- **Cryptographic extension** $c_j = Enc_{p_k}(x_j),\ c_{j'} = Enc_{p_k}(x_{j'})$

$$\begin{aligned} c_j &\leftarrow c_j \cdot c_{j'}^{2^{t_j - t_{j'}}} \quad \text{if } t_j \geq t_{j'}, \\ c_j &\leftarrow c_{j'} \cdot c_j^{2^{t_{j'} - t_j}} \quad \text{otherwise.} \end{aligned}$$

  - **Convergence:** $c_j \rightarrow \mathrm{Enc}_{p_k}(2^T \mu)$ as $t \rightarrow \infty$

Cryptographic Gossip-based aggregation

- **Update**

$$\mu_j \leftarrow \frac{\mu_j + \mu_{j'}}{2},$$

- **Asynchronous update without division**

$$\begin{aligned} \mu_j &\leftarrow \mu_j + 2^{t_j - t_{j'}}\,\mu_{j'}, \quad \text{if } t_j \geq t_{j'} \\ \mu_j &\leftarrow \mu_{j'} + 2^{t_{j'} - t_j}\,\mu_j, \quad \text{otherwise.} \end{aligned}$$

- **Cryptographic extension** $c_j = Enc_{p_k}(x_j),\ c_{j'} = Enc_{p_k}(x_{j'})$

$$\begin{aligned} c_j &\leftarrow c_j \cdot c_{j'}^{2^{t_j - t_{j'}}} \quad \text{if } t_j \geq t_{j'}, \\ c_j &\leftarrow c_{j'} \cdot c_j^{2^{t_{j'} - t_j}} \quad \text{otherwise.} \end{aligned}$$

Cluster centers can be estimated privately!

Private Cluster Label Determination

Server : key pair (p_k,s_k) **Node j** $\boldsymbol{x}_j$,
$(\mathrm{Enc}_{p_k}(2^T\mu_1^1), ..., \mathrm{Enc}_{p_k}(2^T\mu_d^1))$
$(\mathrm{Enc}_{p_k}(2^T\mu_2^1), ..., \mathrm{Enc}_{p_k}(2^T\mu_d^2))$

Problem: which is greater $d(\boldsymbol{x}_j, \boldsymbol{\mu}_1)$ or $d(\boldsymbol{x}_j, \boldsymbol{\mu}_2)$

$w \leftarrow \mathrm{Enc}_{p_k}(d(2^T\boldsymbol{x}_j, 2^T\boldsymbol{\mu}^1) - d(2^T\boldsymbol{x}_j, 2^T\boldsymbol{\mu}^2) - r^B)$

$r^A \leftarrow \mathrm{dec}_{s_k}(w)$
$= d(2^T\boldsymbol{x}_j, 2^T\boldsymbol{\mu}^1) - d(2^T\boldsymbol{x}_j, 2^T\boldsymbol{\mu}^2) - r^B$

r_A and r_B are random shares of
$d(2^T\boldsymbol{x}_j, 2^T\boldsymbol{\mu}^1) - d(2^T\boldsymbol{x}_j, 2^T\boldsymbol{\mu}^2)$

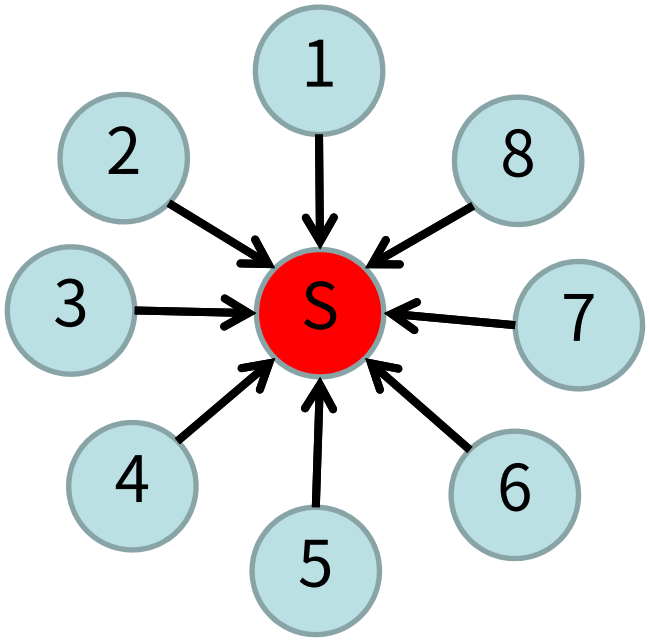
r_A → Comparison by SFE ← r_B

Comparison by SFE → void

Comparison by SFE → The nearest cluster center

The bottleneck of label determination

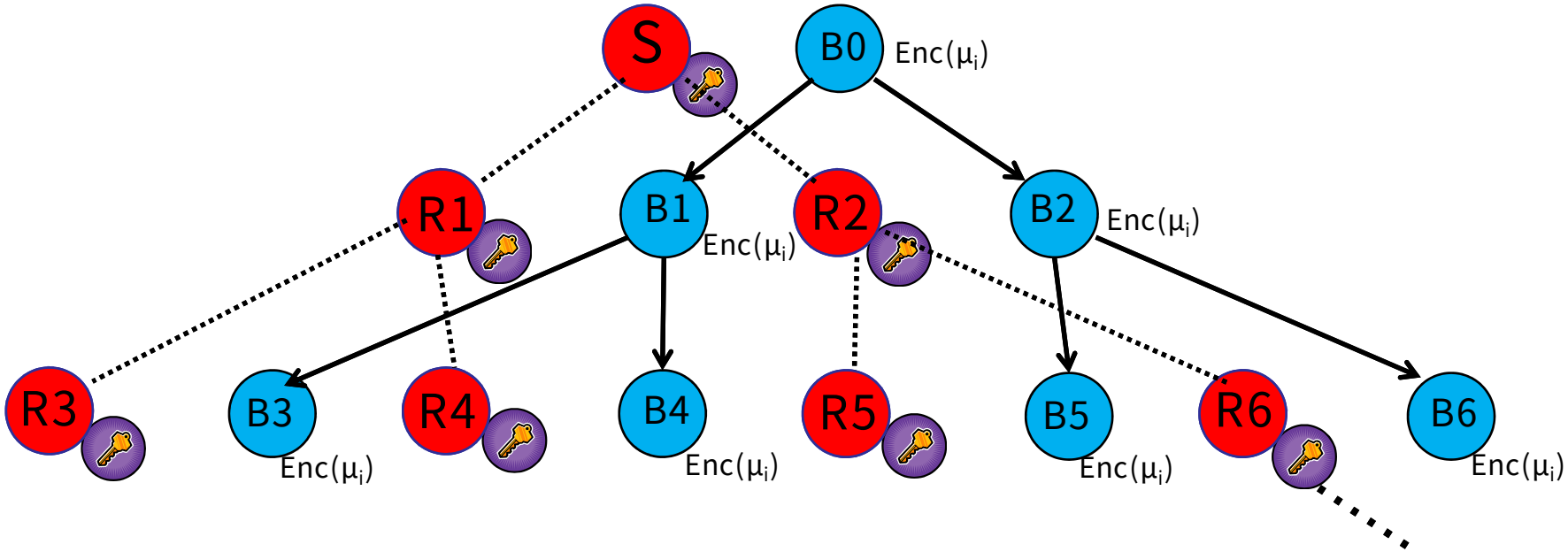
- All nodes need to run the protocol between the server



Bottleneck!

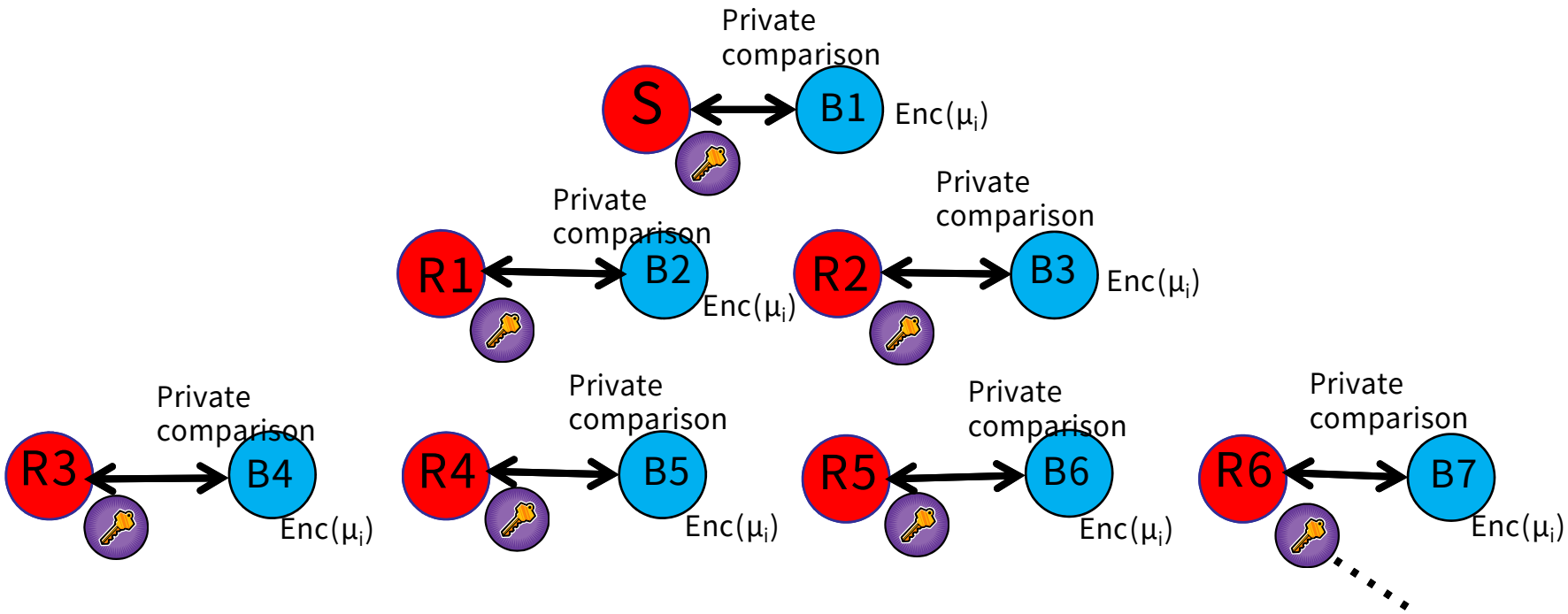
Distributed private update of cluster center

- Nodes Rj and Bj are called “pair”
- $Enc(\mu_i)$ are propagated through the binary tree such that $Enc(\mu_i)$ of Bj can be decrypted only by Rj’s secret key



Distributed private update of cluster center

- Private comparison is processed between “paired” red node Rj and blue node Bj



Privacy-preserving P2P k-means clustering

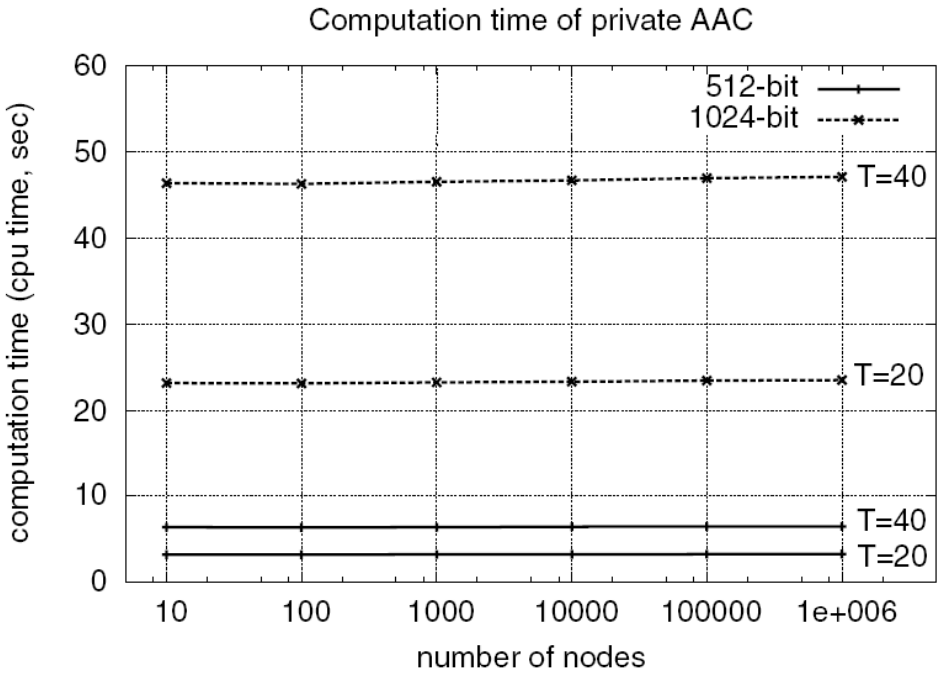
- **Computational Complexity**

	w/o decentralization	with decentralization
Private cluster center update	O(dkT)	---
Private cluster label update	O(dkn)	O(dklogn)

- Total complexity per iteration: O(dkT+dklogn)
- T: maximum cycle of gossip-based aggregation
- n: number of nodes

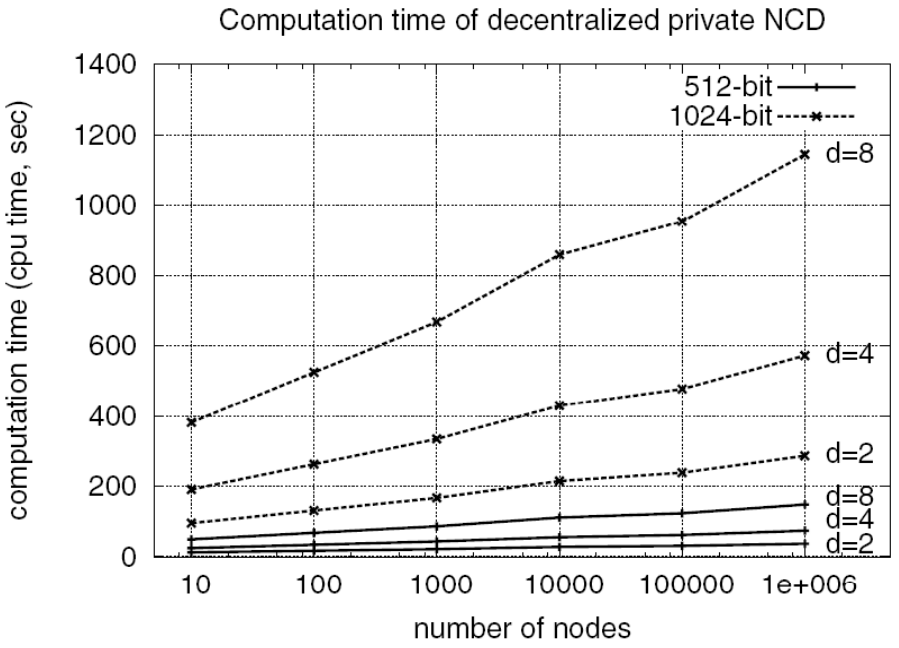
Experiments (private cluster center update)

- **Computational time**
  - # of nodes n=10, ..., 1,000,000
  - Dimension d=1
  - maximum cycle of gossip-based aggreagation T=20,40
  - Paillier cryptosystem [Pai00]
  - Key-length 512bit, 1024bit



Experiments (private cluster labal update)

- **Computational time**
  - # of nodes n=10, …, 1,000,000
  - Dimension d=2,4,8
  - # of clusters k=2
  - Paillier cryptosystem [Paillier00]
  - Key-length 512bit, 1024bit



Experiments (k-means)

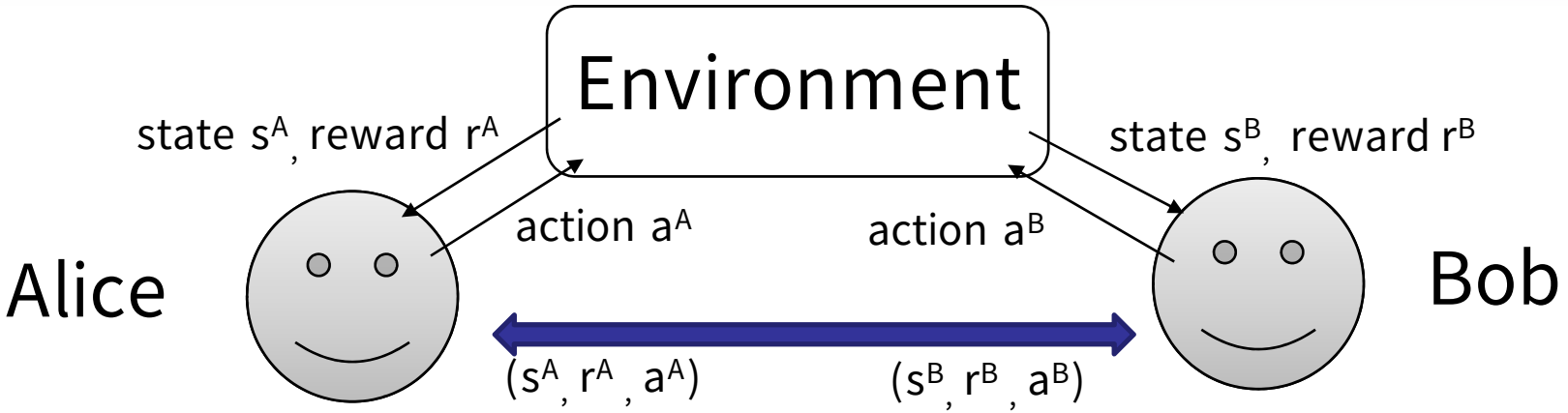
- **Computational time per one step of k-means**

	#of dim	#of clusters	#of nodes	Time (sec) Cluster center	Time (sec) Cluster label
Small scale	2	2	1,000	180	660
Large scale	4	4	1,000,000	740	9,100

- Small scale: about 13(min) per one step
- Large scale: about 2.7 (hour) per one step
- Not very efficient, but can be terminated in practical time

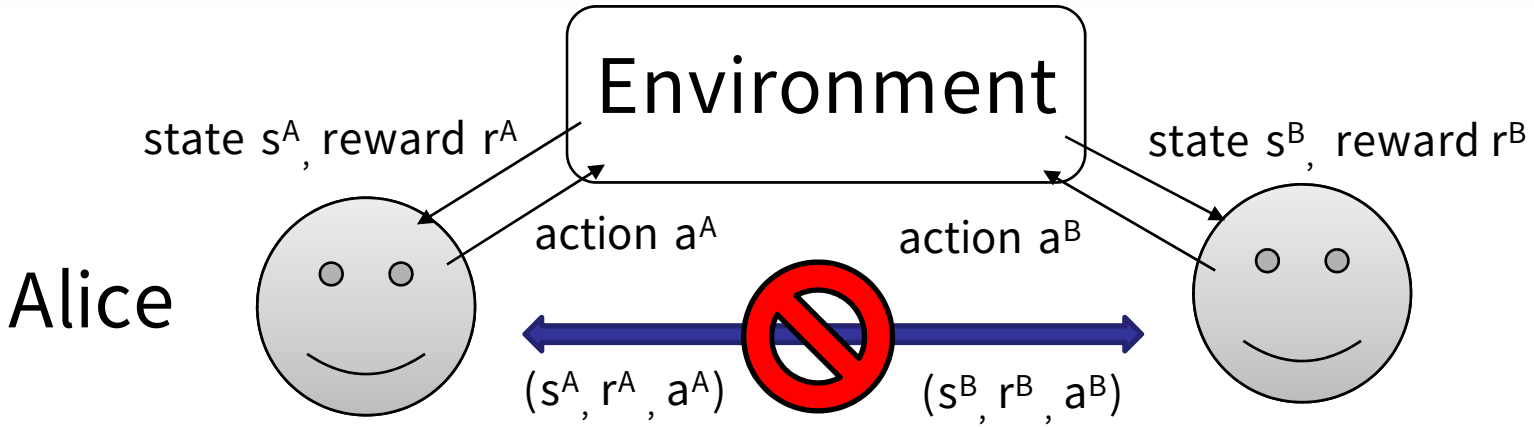
Privacy-preserving Reinforcement Learning

Distributed Reinforcement Learning



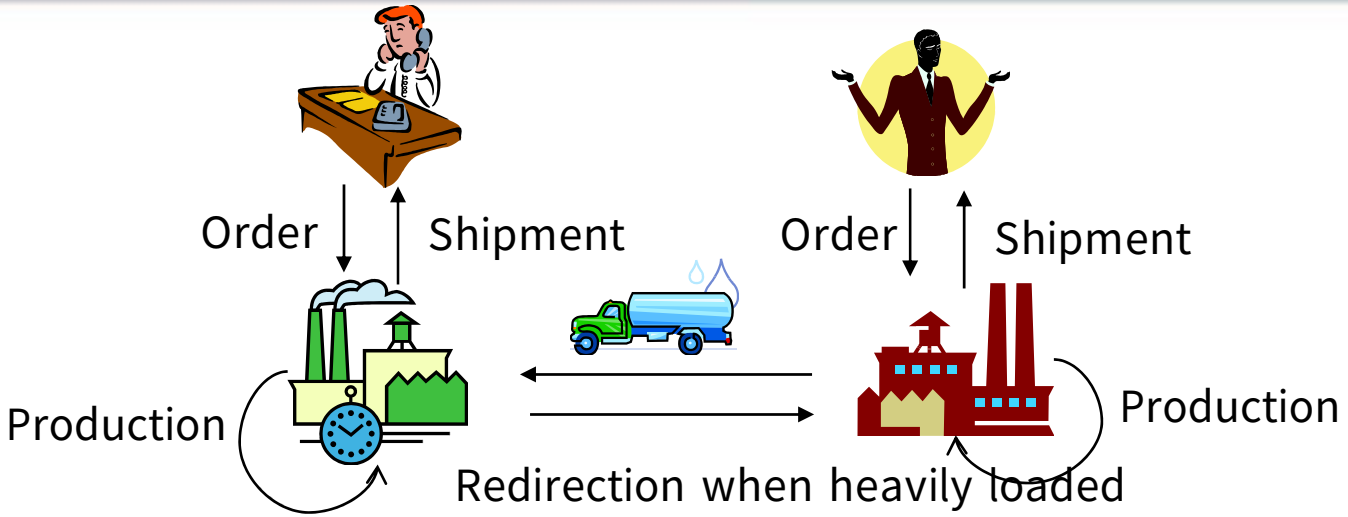
- **Existing approaches for distributed reinforcement learning**
  - Distributed Value Function [Schneider99]
  - Policy gradient approach [Peshkin00][Moallemi03][Bagnell05]
- **The main focus**
  - Manage huge state-action space
  - Limit the communication

DRL from private perceptions



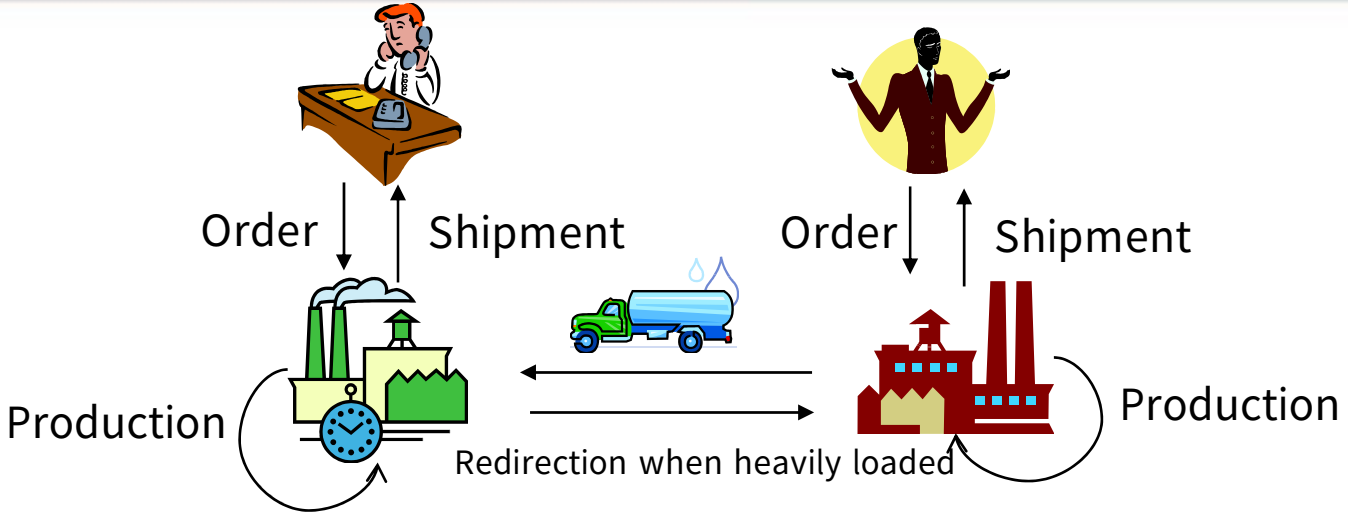
- **DRL from private perceptions**
  - Agents have sufficient computation resources and communication bandwidth
  - Their perceptions are desired to be kept private
    - Alice does not wish to reveal (s^A, r^A, a^A) to Bob
    - Bob does not wish to reveal (s^B, r^B, a^B) to Alice, either
  - In the end, Alice and Bob wish to learn the optimal policy collaboratively

Motivating application: Load balancing



- **A load balancing among competing factories**
  - Obtain a reward by processing a job, but
  - Factories may need to redirect jobs to the other factory when heavily loaded
    - Large penalty for overflow
    - Small penalty for redirection
- **When should factories redirect jobs to the other factory?**

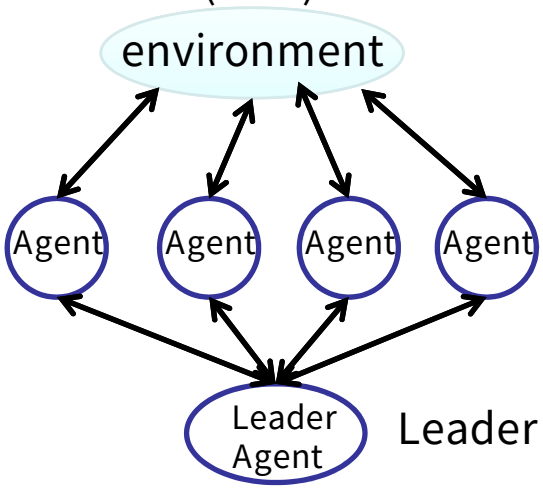
Motivating application: Load balancing



- **If two factories are competing…**
  - The frequency of orders and the speed of production is private (private model)
  - The backlog is private (private state observation)
  - The profit is private (private reward)
- **Privacy-preserving Reinforcement Learning**
  - States, actions, and rewards are not shared
  - But the learned policy is shared in the end

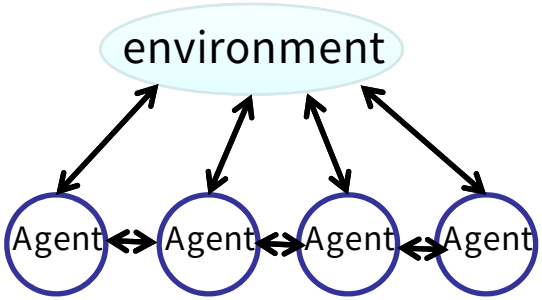
Are existing RLs privacy-preserving?

Centralized RL (CRL)



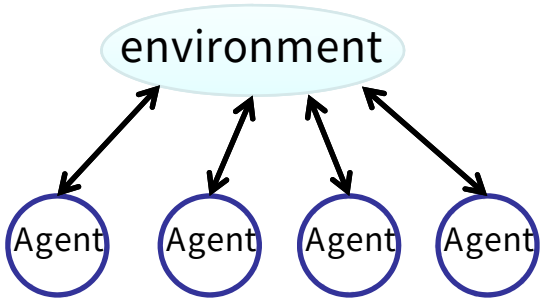
Leader agent learns

Distributed RL (DRL), [Schneider99][Ng05]



Each distributed agent shares partial observation and learns

Independent DRL (IDRL)



Each agent learns independently

	Optimality	Privacy
CRL	optimal	disclosed
DRL	medium	partly disclosed
IDRL	bad	Preserved
PPRL	**optimal**	**preserved**

Target: achievement of privacy preservation without sacrificing the optimality

Step 3: Private Update of Q-values

$$\begin{aligned} \Delta Q(s_t, a_t) &\leftarrow \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)), \\ Q(s_t, a_t) &\leftarrow \Delta Q(s_t, a_t) + Q(s_t, a_t), \qquad (1) \end{aligned}$$

×K, ×L

$$K\Delta Q(s_t, a_t) \leftarrow K\alpha(Lr_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

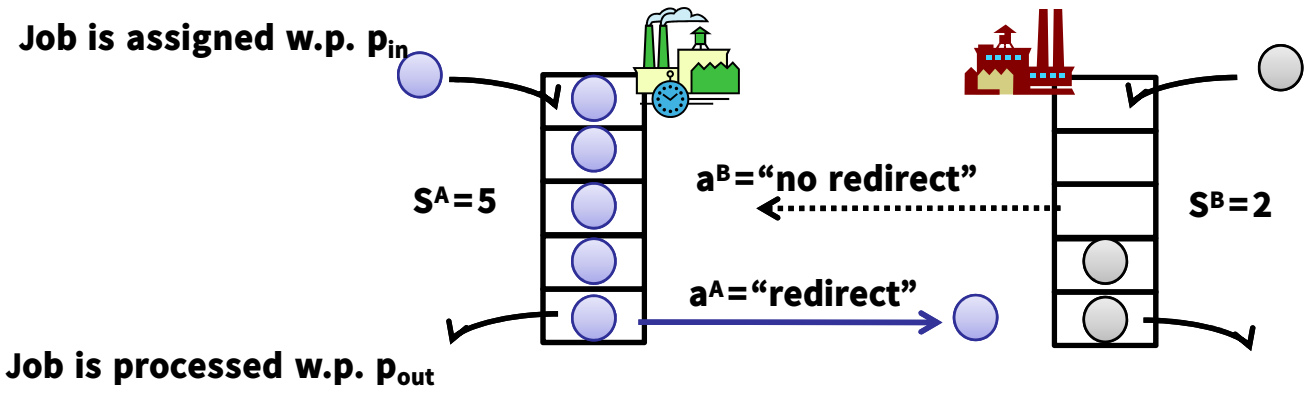
Encryption

$$\begin{aligned} &e_{\mathsf{pk}_A}(K\Delta Q(s_t, a_t)) \\ &\quad = e_{\mathsf{pk}_A}(Lr_t)^{\alpha K} \cdot c(s_{t+1}, a_{t+1})^{\alpha\gamma K} \cdot c(s_t, a_t)^{-\alpha K} \end{aligned}$$

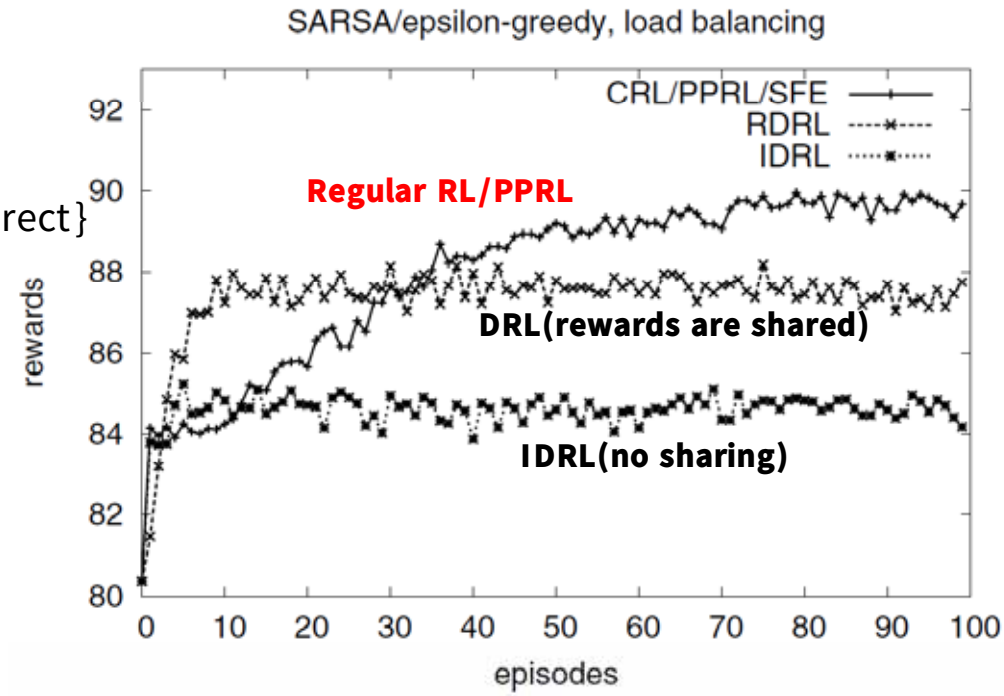
$$c(s_t, a_t) \leftarrow e_{\mathsf{pk}_A}(K\Delta Q(s_t, a_t)) \cdot c(s_t, a_t)$$

The agent can update c(s,a) without knowledge of Q(s,a)!

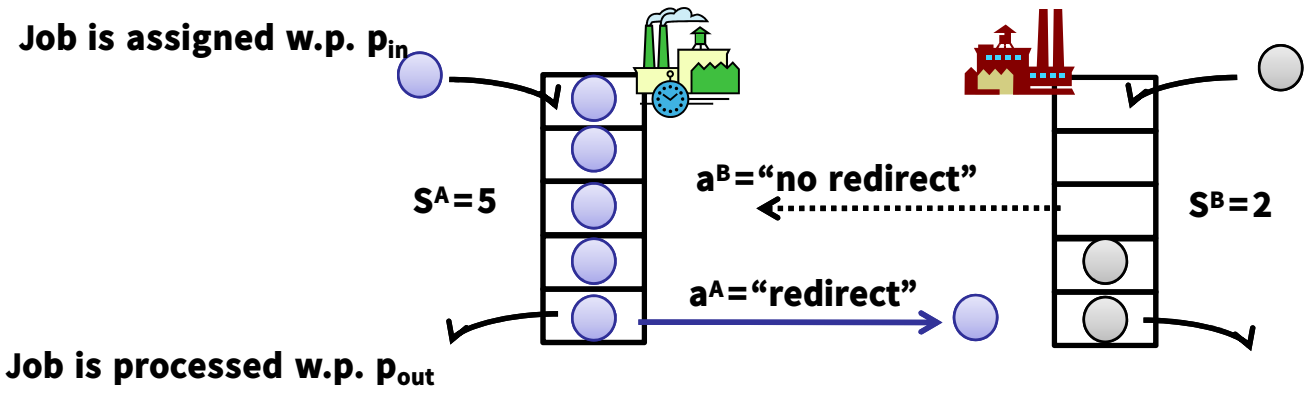
Experiment: Load balancing among factories



- **Setting**
  - State space: $S^A, S^B \in \{0,1,\cdots,5\}$
  - Action space: $A^A, A^B \in$ {redirect, no redirect}
  - Reward:
    - Cost for backlog : $r^A = 50-(s^A)^2$
    - Cost for redirection: $r^A = r^A - 2$
    - Cost for overflow: $r^A = 0$
      - Reward r^B is set similarly
    - System reward: $r_t = r_t^A + r_t^B$



Experiment: Load balancing among factories



- **Comparison**

* Java 1.5.0+Fairplay, 1.2 GHz Core solo

	detail	comp(sec)*	profit	privacy
CRL	All information shared	5.11	**90.0**	Disclosed all
DRL	Rewards are shared	5.24	87.4	Partially disclosed
IDRL	No sharing	5.81	84.2	**Perfect**
PPRL	***Security protocol***	**$8.85*10^5$**	**90.0**	**Perfect**
RL/SFE	SFE[Yao86]	$>7.00*10^7$	**90.0**	**Perfect**

Future direction

- **Many privacy-preserving version of data mining and machine learning have been presented**
  - Classifier: Decision tree, naïve Bayes, support vector machine, k-nearest neighbor
  - Clustering: k-means, EM for mixture models, DBSCAN
  - Machine learning: Linear regression, Bayesian network construction(k2), brief propagation, boosting, reinforcement learning
- **Next step**
  - Connection with personalized services
    - To increase the degree of personalization, more sensitive information will be required
  - Connection with network/graph mining
    - links in network are intrinsically private
    - E.g., Personal / business network, Communication graph
  - Connection with ubiquitous appliances and environment: mining with…
    - Cell phone + geographical movement
    - RFID attached to personal items
  - Application:
    - Planned in information grand voyage project

多様な情報価値を共創する
情報大航海プロジェクト