PRACTICAL PRIVACY-PRESERVING K-MEANS CLUSTERING

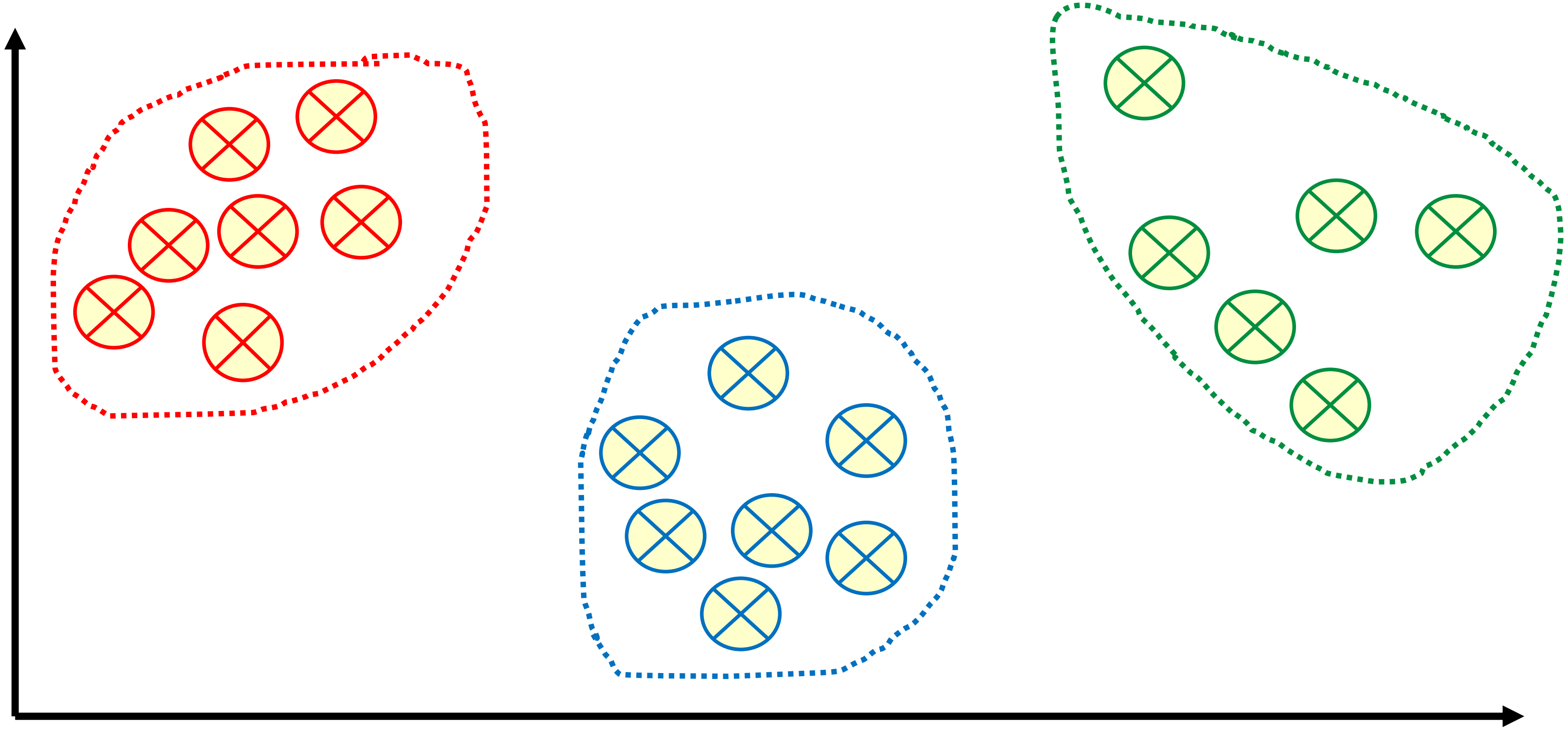
Recording: https://www.youtube.com/watch?v=g7292mN-yAI

1

Payman Mohassel (Facebook)
Mike Rosulek (Oregon State University)
Ni Trieu (UC Berkeley)

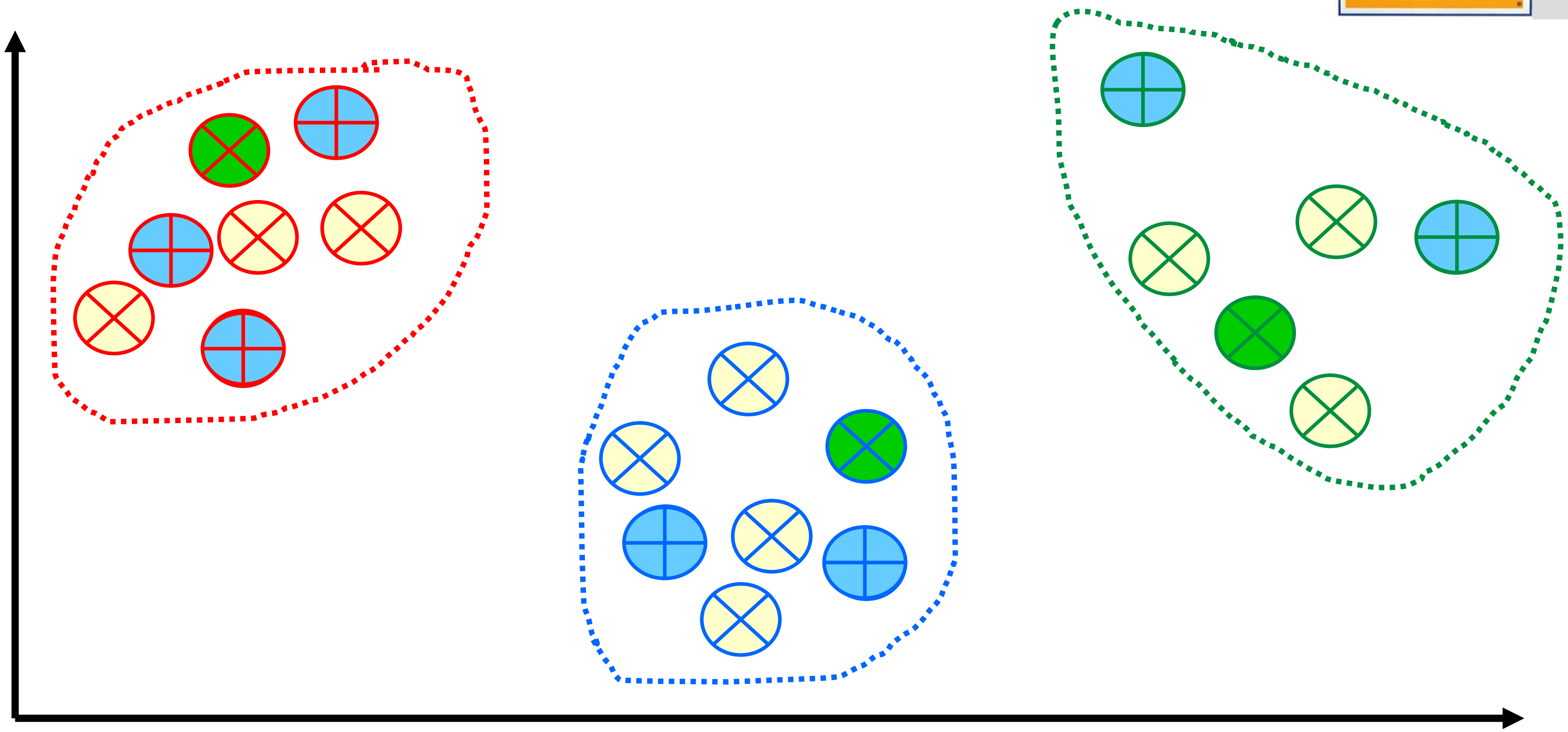
~~WHY DO PRIVACY PRESERVING~~ CLUSTERING?

- What is clustering?
  - The process of grouping a set of objects into classes of similar objects

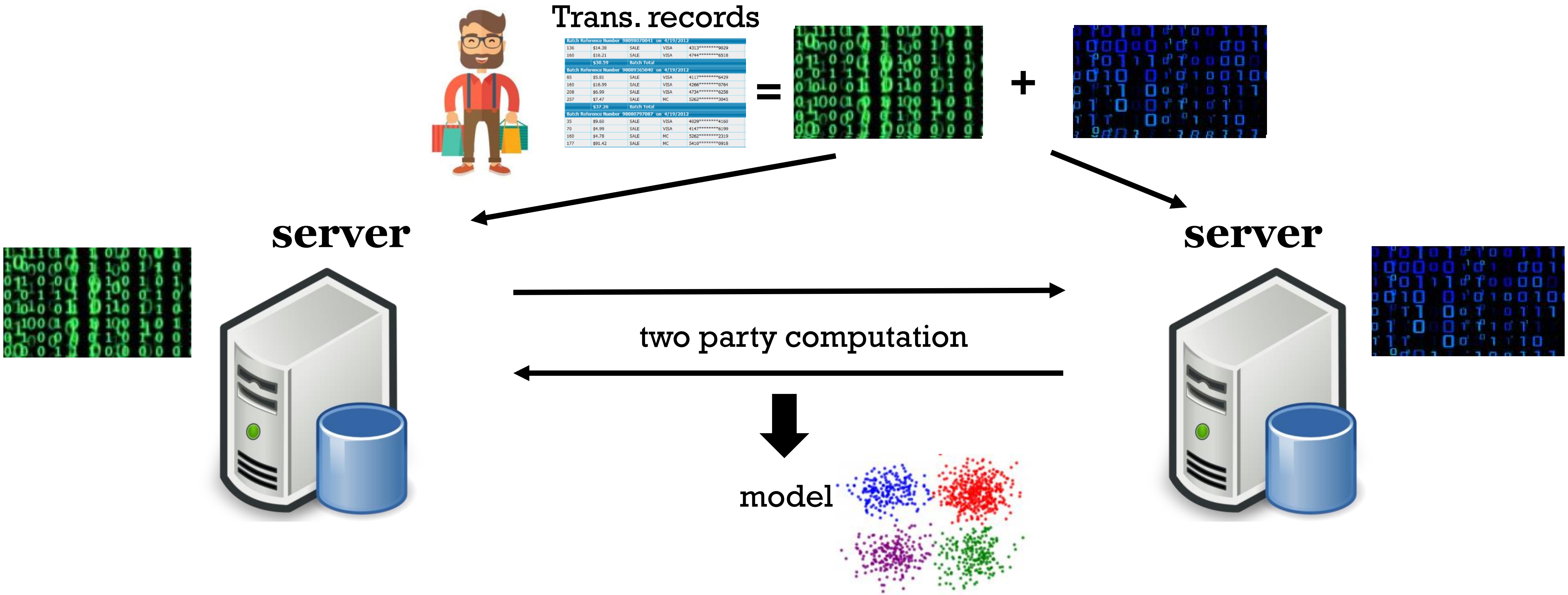


WHY DO PRIVACY PRESERVING CLUSTERING?

- What is clustering?
  - The process of grouping a set of objects into classes of similar objects
- Why is privacy preserving clustering important?
  - Data comes from different sources: transaction, genome
  - Privacy concern



THIS WORK: TWO-SERVER MODEL

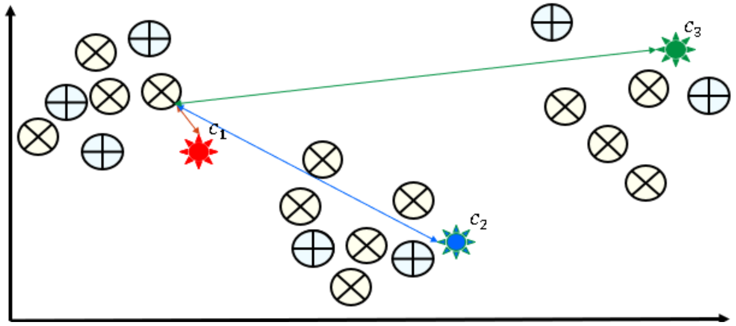


- Our scheme:
  - More efficient than previous work
  - Scale to large datasets

CLUSTERING ALGORITHMS

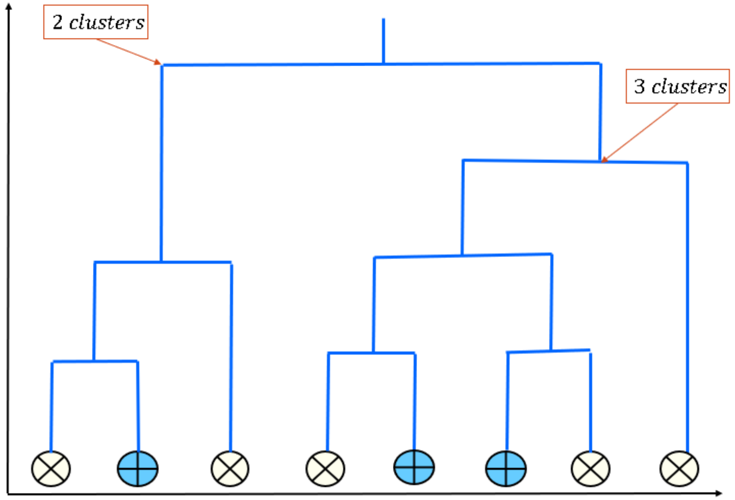
- K-mean clustering $O(nmt)$, where dataset size n, number clusters m, number iterations t

This work



$m \ll n$ and t $\ll n$

- Hierarchical clustering $O(n^2 \log n)$



- K-mean clustering is faster than hierarchical clustering
- K-mean clustering is widely used in many applications

PRIVACY-PRESERVING K-MEAN CLUSTERING

- Privacy-preserving K-mean Clustering has received less attention than other supervised ML problems
- Previous work […,XHYCZ17, JA18]
  - Not provide a full privacy guarantee (e.g. reveal the intermediate cluster centers)
  - and/or heavy on public key operations

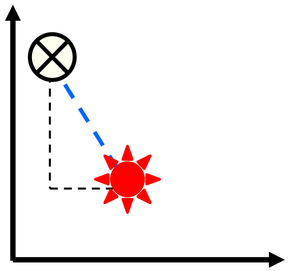
  $\Rightarrow$ inefficient when the dataset is large.
- Our work:
  - Full privacy guarantee
  - Based on symmetric key operations
    - Efficient secure Euclidean distance
    - Customized minimum circuit on shared values
    - K-means clustering

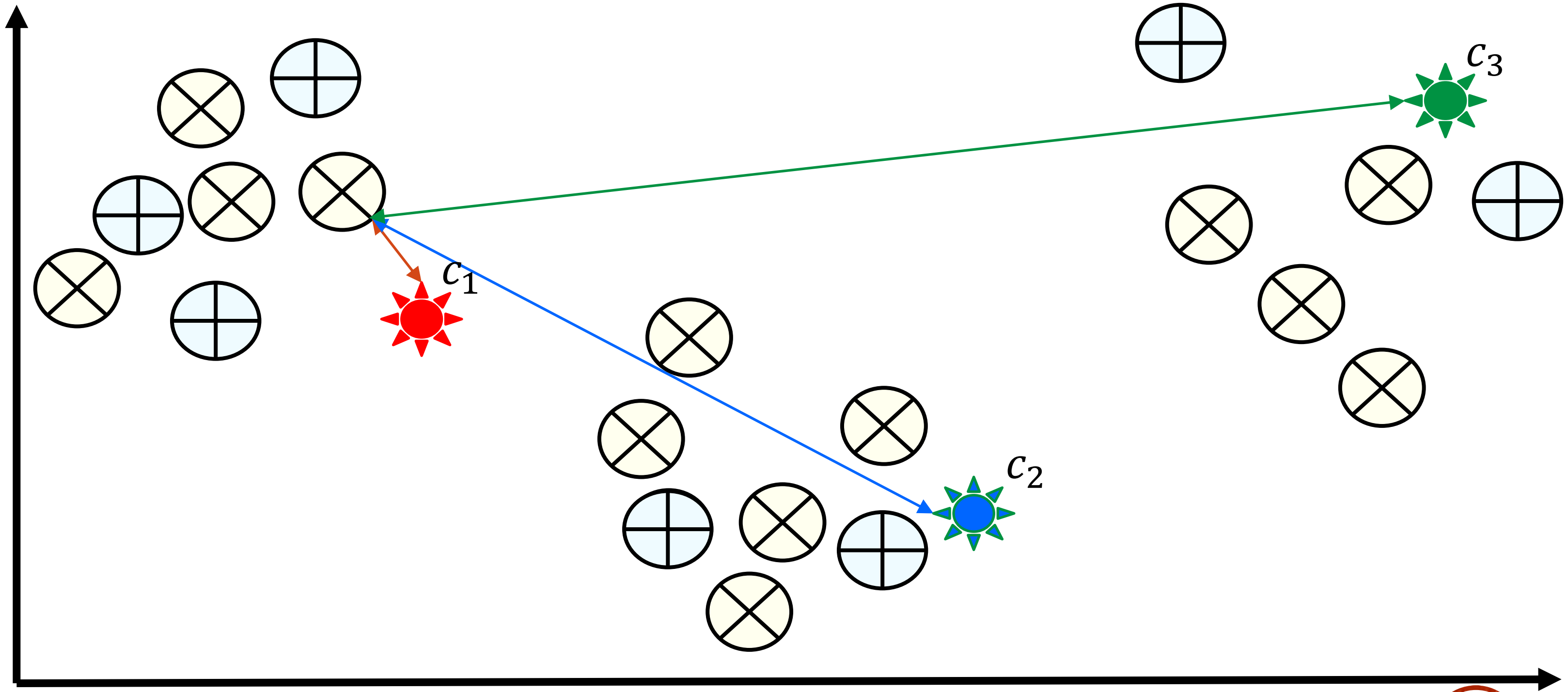
  $\Rightarrow$ 5 orders of magnitude faster than prior work

PRIVACY-PRESERVING K-MEANS CLUSTERING

- Inputs:
  - Secret shared dataset as $\{[d_1], \dots, [d_n]\}$
  - #clusters m
- Algorithm:
  - Pick m random clusters $\{[c_1], \dots, [c_m]\}$
  - Until clustering *converges* (or other stopping criterion):
    - For each data point $\boldsymbol{d_i}$:
      - For each cluster $\boldsymbol{c_k}$:
        - Compute the distance between $\boldsymbol{d_i}$ and $\boldsymbol{c_k}$ as $[e_{ik}]$ =dist($[d_i]$, $[c_k]$)

dist($[d_i]$, $[c_k]$)

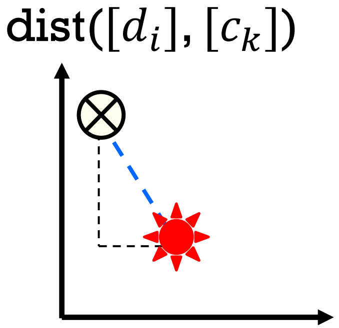


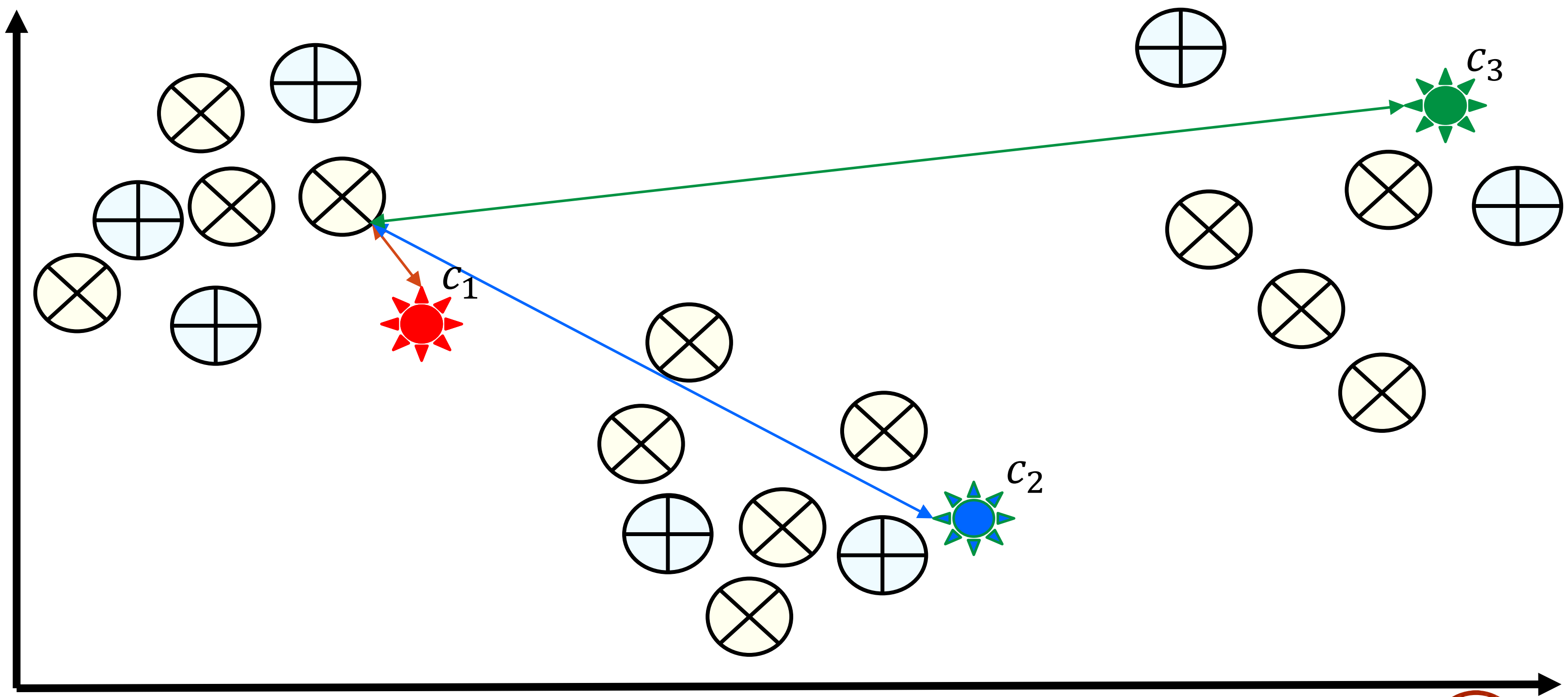


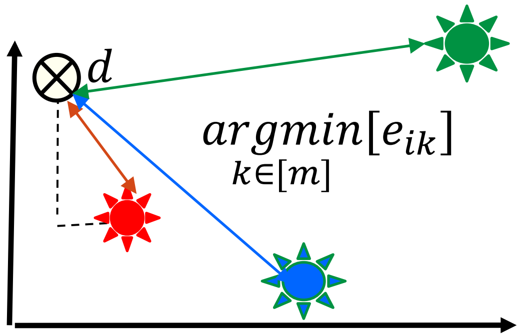
PRIVACY-PRESERVING K-MEANS CLUSTERING

- Inputs:
  - Secret shared dataset as $\{[d_1], \ldots, [d_n]\}$
  - #clusters m
- Algorithm:
  - Pick m random clusters $\{[c_1], \ldots, [c_m]\}$
  - Until clustering *converges* (or other stopping criterion):
    - For each data point $\boldsymbol{d_i}$:
      - For each cluster $\boldsymbol{c_k}$:
        - Compute the distance between $\boldsymbol{d_i}$ and $\boldsymbol{c_k}$ as $[e_{ik}]$ =dist$([d_i], [c_k])$
      - Find closest cluster center

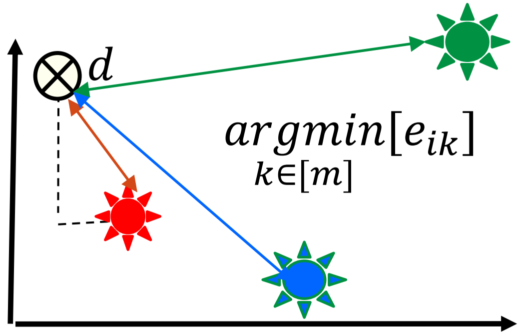
$$[\boldsymbol{k}^*] = \underset{\boldsymbol{k} \in [\boldsymbol{m}]}{\boldsymbol{argmin}}[\boldsymbol{e_{ik}}]$$

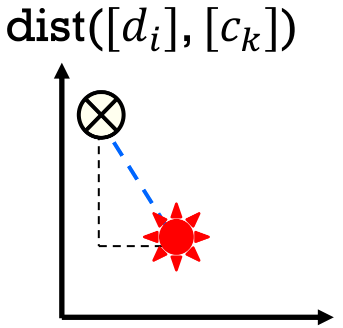






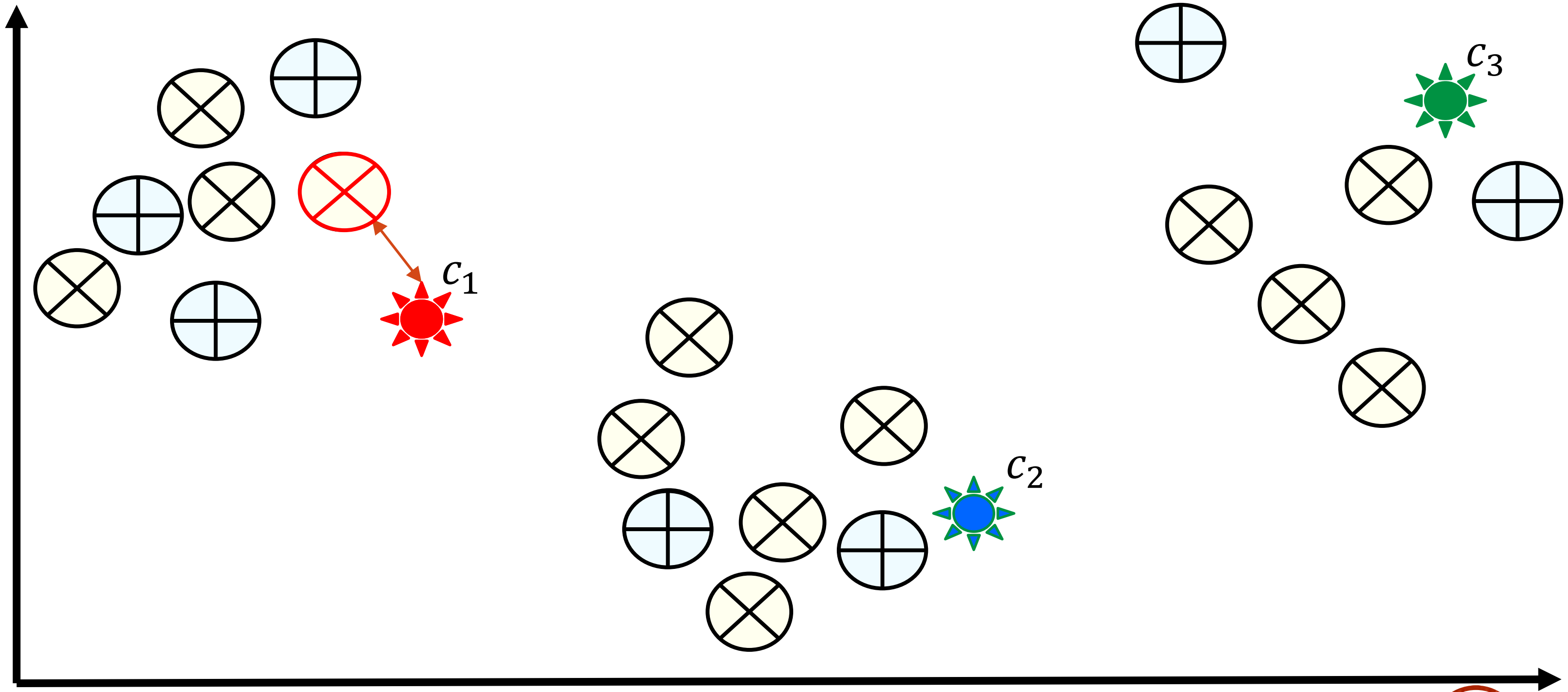
PRIVACY-PRESERVING K-MEANS CLUSTERING



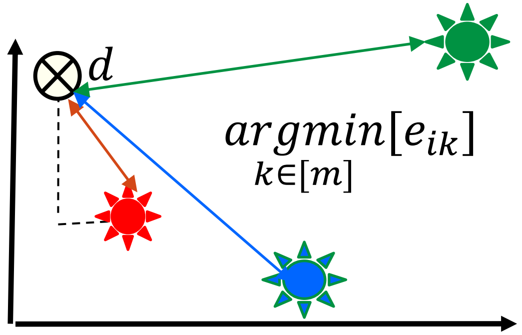


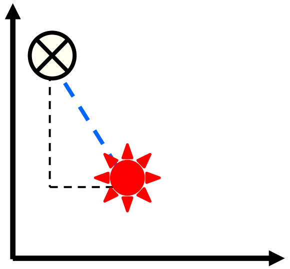
- Inputs:
  - Secret share dataset as $\{[d_1], \dots, [d_n]\}$
  - #clusters m
- Algorithm:
  - Pick m random clusters $\{[c_1], \dots, [c_m]\}$
  - Until clustering *converges* (or other stopping criterion):
    - For each data point $\boldsymbol{d_i}$:
      - For each cluster $\boldsymbol{c_k}$:
        - Compute the distance between $\boldsymbol{d_i}$ and $\boldsymbol{c_k}$ as $[e_{ik}]$ =dist($[d_i]$, $[c_k]$)
      - Find closest cluster center

        $[\boldsymbol{k}^*] = \underset{\boldsymbol{k}\in[\boldsymbol{m}]}{\boldsymbol{argmin}}[\boldsymbol{e_{ik}}]$
      - Assign the data point $\boldsymbol{d_i}$ to the closest cluster center $\boldsymbol{k}^*$



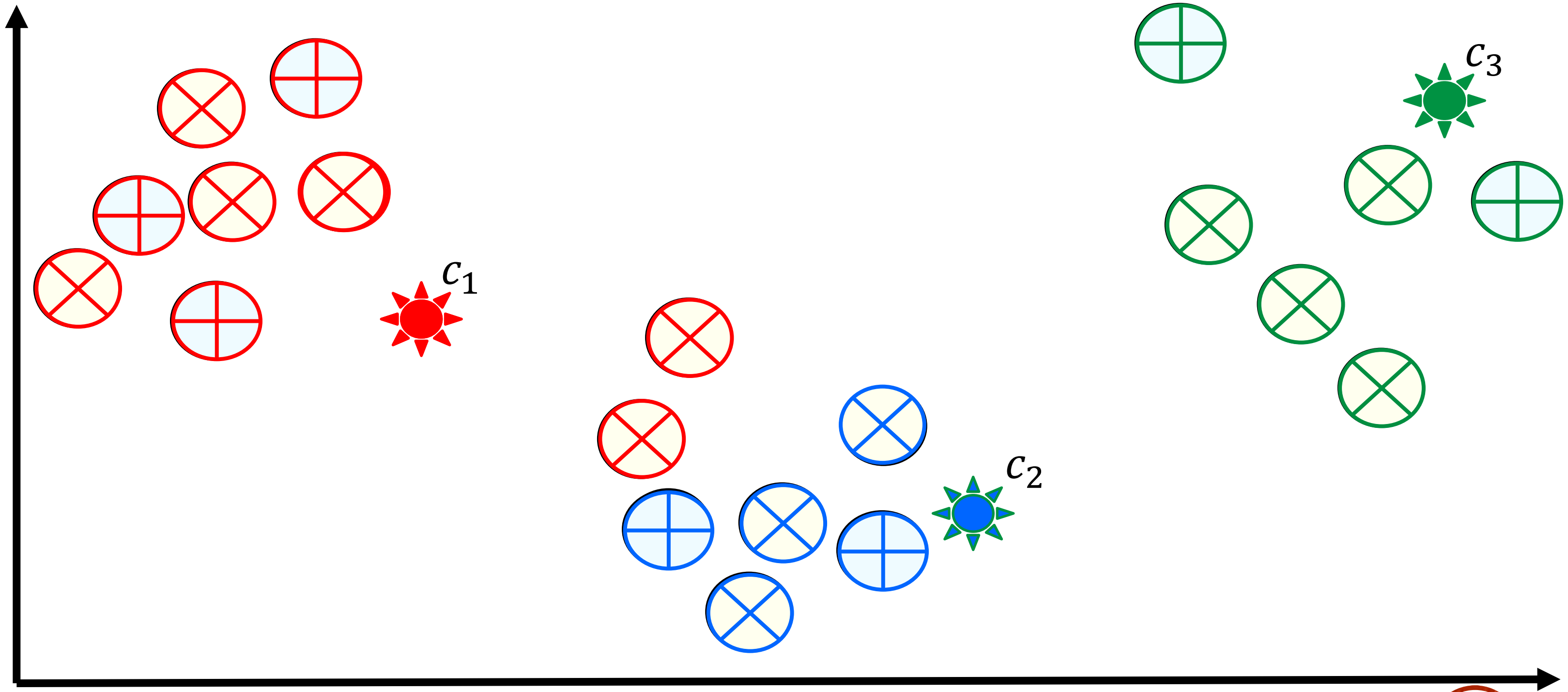
PRIVACY-PRESERVING K-MEANS CLUSTERING



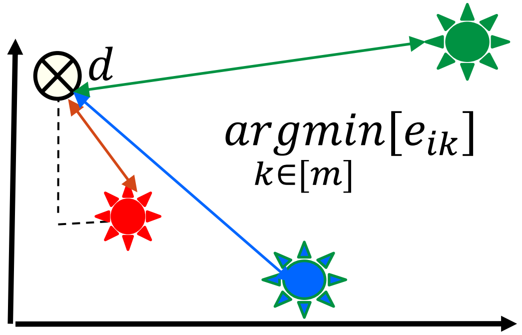


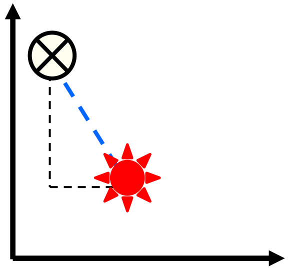
- Inputs:
  - Secret share dataset as $\{[d_1], \dots, [d_n]\}$
  - #clusters m
- Algorithm:
  - Pick m random clusters $\{[c_1], \dots, [c_m]\}$
  - Until clustering *converges* (or other stopping criterion):
    - For each data point $\boldsymbol{d_i}$:
      - For each cluster $\boldsymbol{c_k}$:
        - Compute the distance between $\boldsymbol{d_i}$ and $\boldsymbol{c_k}$ as $[e_{ik}]$ =dist($[d_i], [c_k]$)
      - Find closest cluster center

        $[\boldsymbol{k}^*] = \underset{\boldsymbol{k\in[m]}}{\boldsymbol{argmin}}[\boldsymbol{e_{ik}}]$
      - Assign the data point $\boldsymbol{d_i}$ to the closest cluster center $\boldsymbol{k}^*$



PRIVACY-PRESERVING K-MEANS CLUSTERING

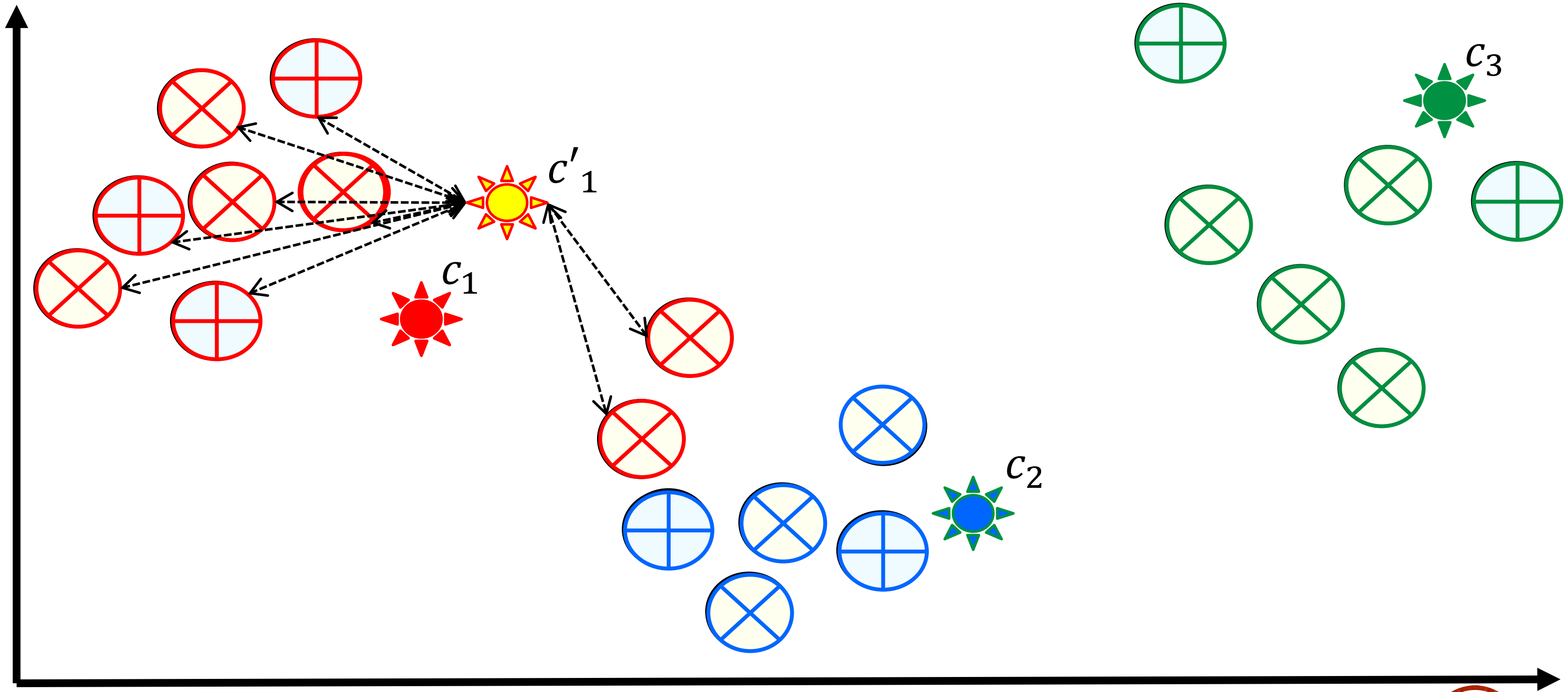




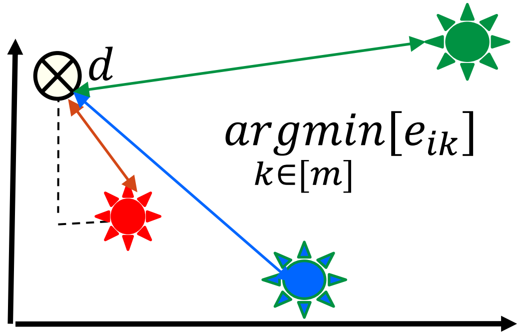
- Inputs:
  - Secret share dataset as $\{[d_1], \ldots, [d_n]\}$
  - #clusters m
- Algorithm:
  - Pick m random clusters $\{[c_1], \ldots, [c_m]\}$
  - Until clustering *converges* (or other stopping criterion):
    - For each data point $\boldsymbol{d_i}$:
      - For each cluster $\boldsymbol{c_k}$:
        - Compute the distance between $\boldsymbol{d_i}$ and $\boldsymbol{c_k}$ as $[e_{ik}] = \text{dist}([d_i], [c_k])$
      - Find closest cluster center

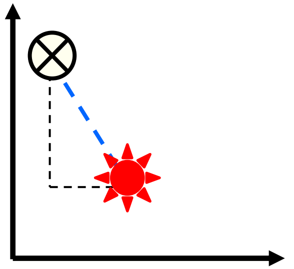
        $$[\boldsymbol{k^*}] = \underset{\boldsymbol{k \in [m]}}{\boldsymbol{argmin}}[\boldsymbol{e_{ik}}]$$
      - Assign the data point $\boldsymbol{d_i}$ to the closest cluster center $\boldsymbol{k^*}$
    - For each cluster $\boldsymbol{c_k}$:
      - Compute the average of the points assigned to the cluster as

        $$[\boldsymbol{c'_k}] = \boldsymbol{avg}([\boldsymbol{d_i}]), \boldsymbol{d_i} \in \boldsymbol{C_k}$$



PRIVACY-PRESERVING K-MEANS CLUSTERING

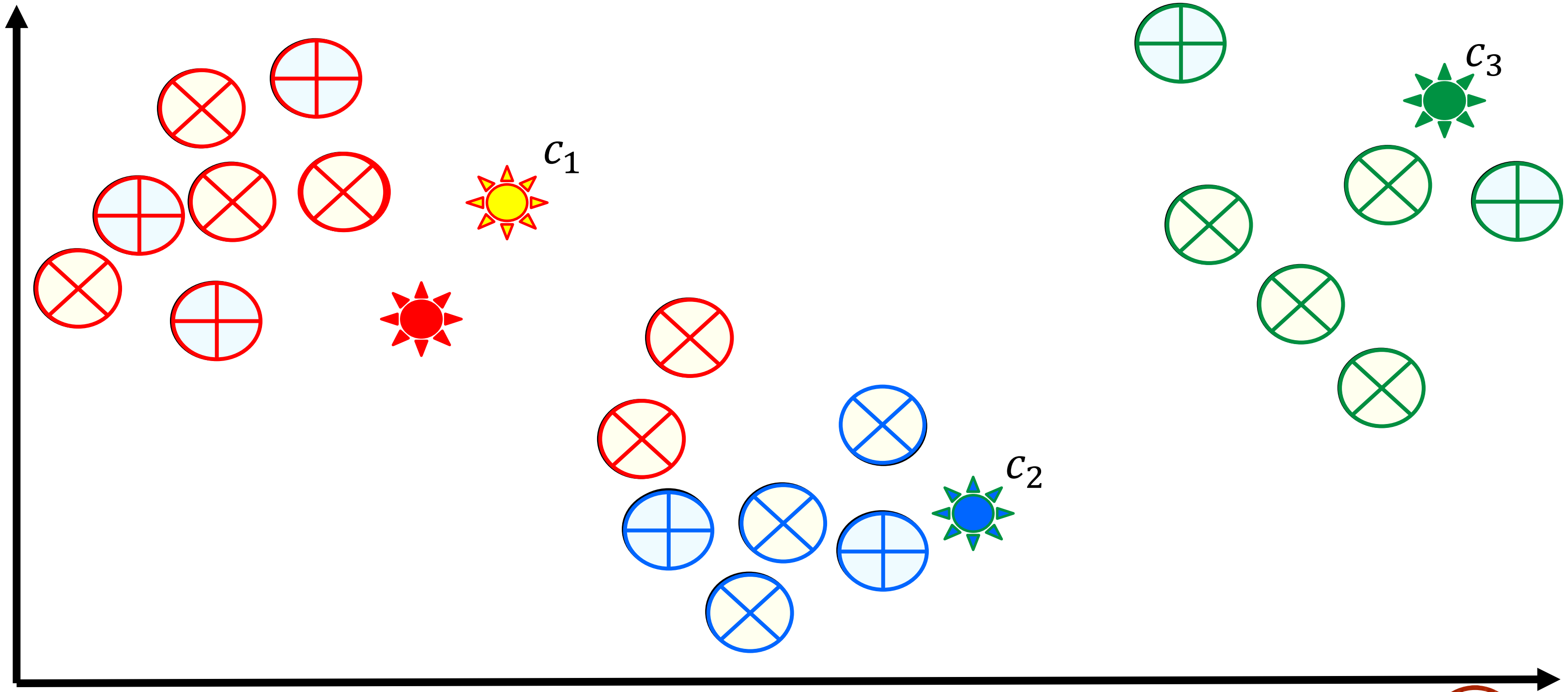




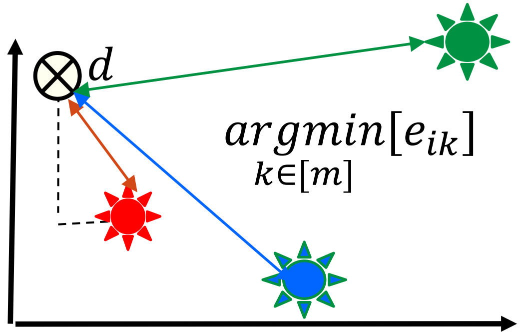
- Inputs:
  - Secret share dataset as $\{[d_1], \dots, [d_n]\}$
  - #clusters m
- Algorithm:
  - Pick m random clusters $\{[c_1], \dots, [c_m]\}$
  - Until clustering *converges* (or other stopping criterion):
    - For each data point $\boldsymbol{d_i}$:
      - For each cluster $\boldsymbol{c_k}$:
        - Compute the distance between $\boldsymbol{d_i}$ and $\boldsymbol{c_k}$ as $[e_{ik}]$ =dist($[d_i]$, $[c_k]$)
      - Find closest cluster center

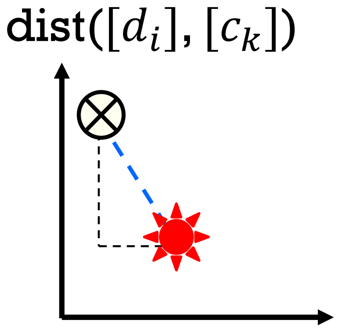
        $[\boldsymbol{k^*}] = \underset{\boldsymbol{k\in[m]}}{\boldsymbol{argmin}}[\boldsymbol{e_{ik}}]$
      - Assign the data point $\boldsymbol{d_i}$ to the closest cluster center $\boldsymbol{k^*}$
    - For each cluster $\boldsymbol{c_k}$:
      - Compute the average of the points assigned to the cluster as

        $[\boldsymbol{c'_k}] = \boldsymbol{avg}([\boldsymbol{d_i}]), \boldsymbol{d_i} \in \boldsymbol{C_k}$
      - Update $\boldsymbol{c_k} = \boldsymbol{c'_k}$



PRIVACY-PRESERVING K-MEANS CLUSTERING

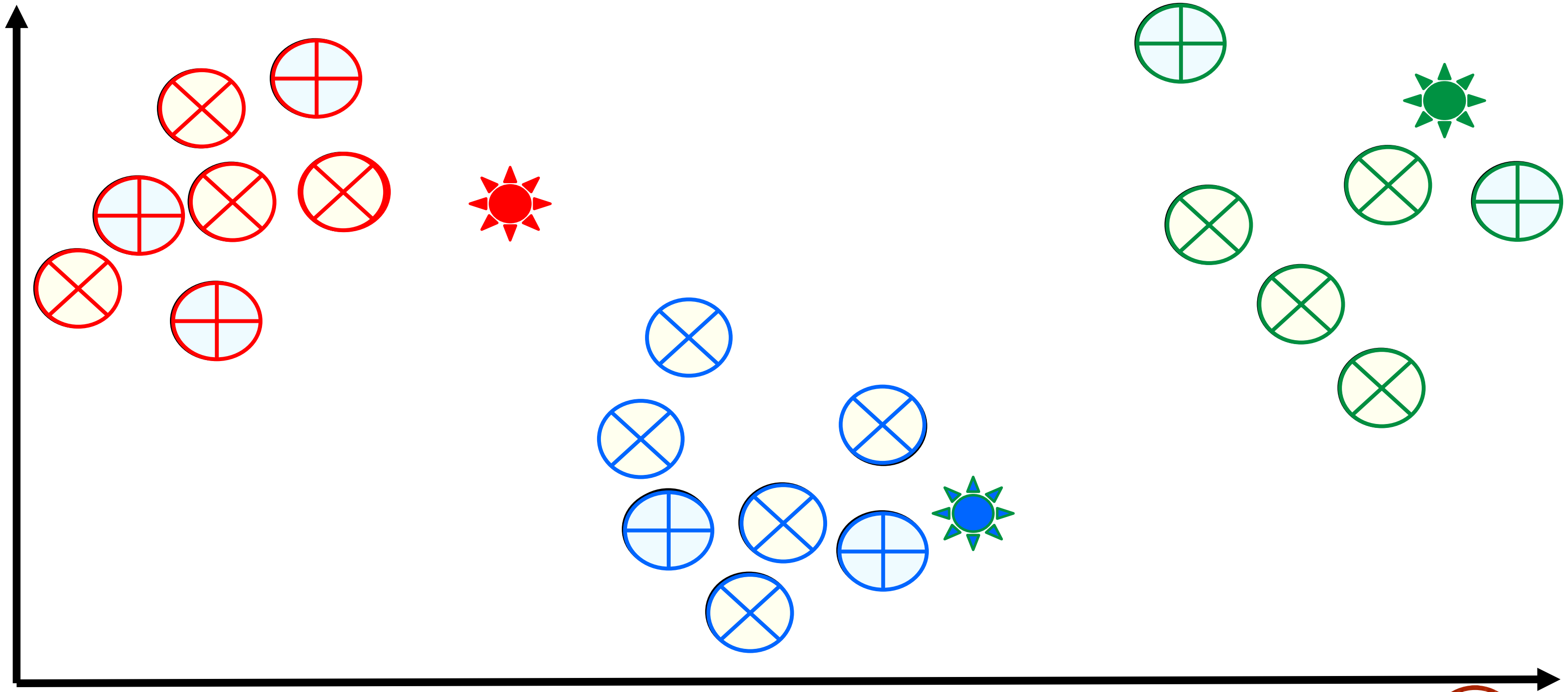




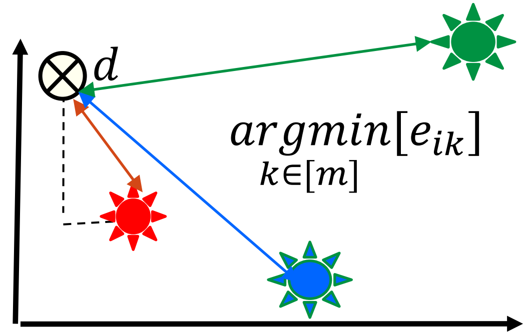
- Inputs:
  - Secret share dataset as $\{[d_1], \ldots, [d_n]\}$
  - #clusters m
- Algorithm:
  - Pick m random clusters $\{[c_1], \ldots, [c_m]\}$
  - Until clustering *converges* (or other stopping criterion):
    - For each data point $\boldsymbol{d_i}$:
      - For each cluster $\boldsymbol{c_k}$:
        - Compute the distance between $\boldsymbol{d_i}$ and $\boldsymbol{c_k}$ as $[e_{ik}]$ =dist($[d_i]$, $[c_k]$)
      - Find closest cluster center

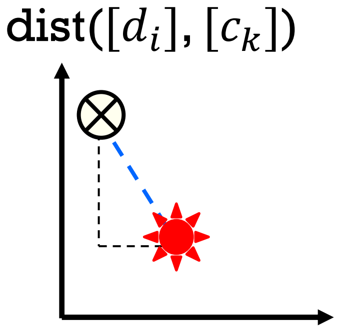
        $$[\boldsymbol{k}^*] = \underset{\boldsymbol{k}\in[\boldsymbol{m}]}{\boldsymbol{argmin}}[\boldsymbol{e_{ik}}]$$
      - Assign the data point $\boldsymbol{d_i}$ to the closest cluster center $\boldsymbol{k}^*$
    - For each cluster $\boldsymbol{c_k}$:
      - Compute the average of the points assigned to the cluster as

        $$[\boldsymbol{c'_k}] = \boldsymbol{avg}([\boldsymbol{d_i}]), \boldsymbol{d_i} \in \boldsymbol{C_k}$$
      - Update $\boldsymbol{c_k} = \boldsymbol{c'_k}$



PRIVACY-PRESERVING K-MEANS CLUSTERING

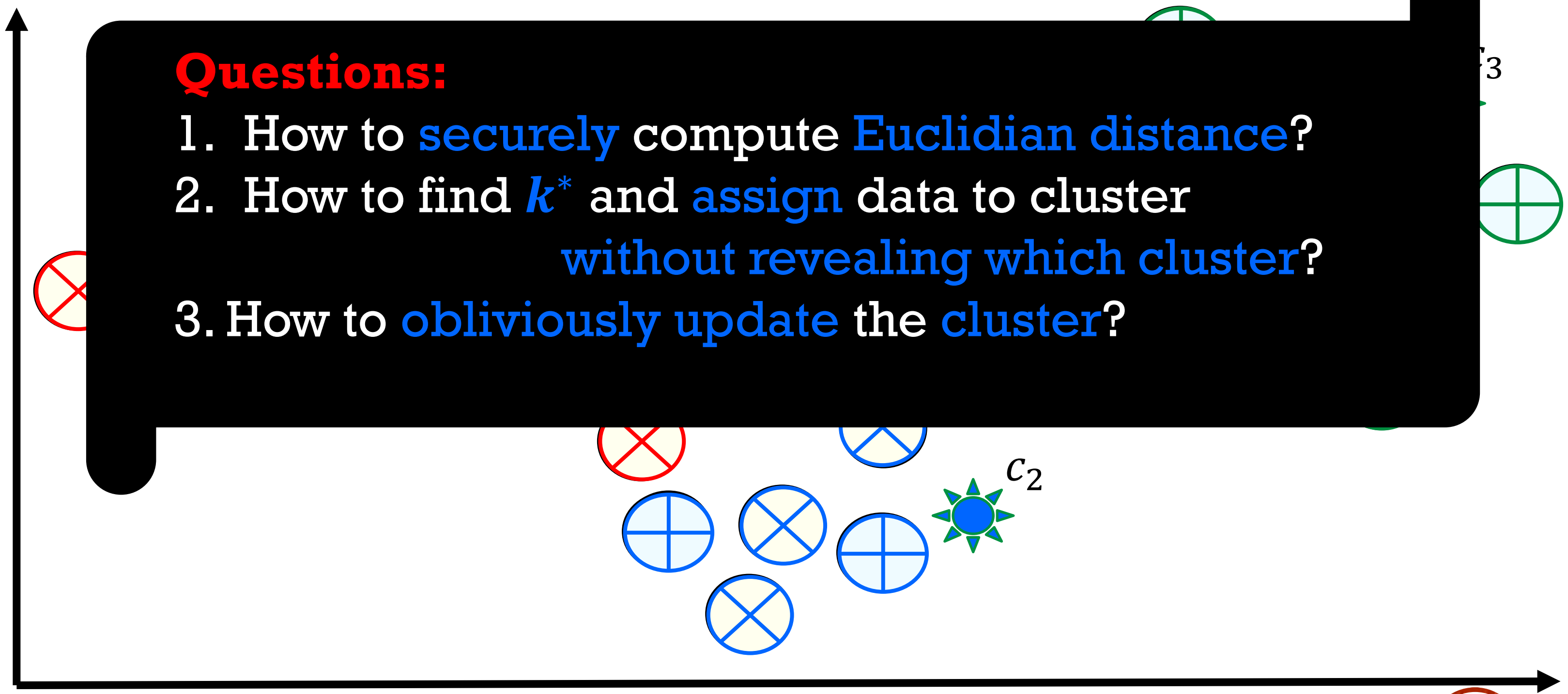




- Inputs:
  - Secret share dataset as $\{[d_1], \dots, [d_n]\}$
  - #clusters m
- Algorithm:
  - Pick m random clusters $\{[c_1], \dots, [c_m]\}$
  - Until clustering *converges* (or other stopping criterion):
    - For each data point $\boldsymbol{d_i}$:
      - For each cluster $\boldsymbol{c_k}$:
        - Compute the distance between $\boldsymbol{d_i}$ and $\boldsymbol{c_k}$ as $[e_{ik}]$ =dist($[d_i], [c_k]$)
      - Find closest cluster center

        $[\boldsymbol{k^*}] = \underset{k\in[m]}{\boldsymbol{argmin}}[\boldsymbol{e_{ik}}]$
      - Assign the data point $\boldsymbol{d_i}$ to the closest cluster center $\boldsymbol{k^*}$
    - For each cluster $\boldsymbol{c_k}$:
      - Compute the average of the points assigned to the cluster as

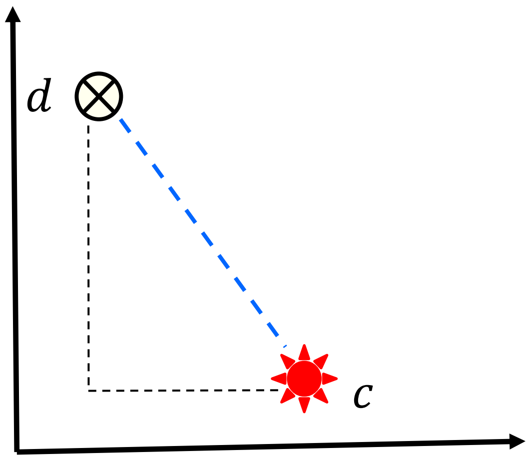
        $[\boldsymbol{c'_k}] = \boldsymbol{avg}([\boldsymbol{d_i}]), \boldsymbol{d_i} \in \boldsymbol{C_k}$
      - Update $\boldsymbol{c_k} = \boldsymbol{c'_k}$



Questions:
1. How to securely compute Euclidian distance?
2. How to find k^* and assign data to cluster without revealing which cluster?
3. How to obliviously update the cluster?

SECURE EUCLIDIAN DISTANCE

dist($[d]$, $[c]$)
$= (d_1 - c_1)^2 + (d_2 - c_2)^2$



- Squared Euclidian distance dist($[d], [c]$) $= \sum_{i=1}^{\ell} (d_i - c_i)^2$, where:
  - d,c is ℓ dimensional vector
  - Each party holds arithmetic share $d_i = d_i^A + d_i^B$, $c_i = c_i^A + c_i^B$
- dist($[d], [c]$) $= \sum_{i=1}^{\ell} \left(d_i^A + d_i^B - c_i^A - c_i^B\right)^2$

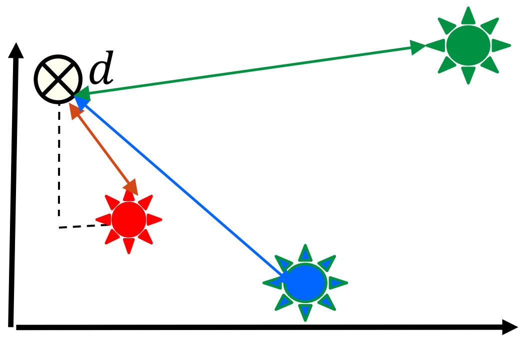
$$= \sum_{i=1}^{\ell} \left((d_i^A - c_i^A) + (d_i^B - c_i^B)\right)^2$$

$$= \sum_{i=1}^{\ell} \left(d_i^A - c_i^A\right)^2 + \sum_{i=1}^{\ell} \left(d_i^B - c_i^B\right)^2 + 2\sum_{i=1}^{\ell} (d_i^A - c_i^A)\,(d_i^B - c_i^B)$$

- Alice locally computes $\sum_{i=1}^{\ell} \left(d_i^A - c_i^A\right)^2$
- Bob locally computes $\sum_{i=1}^{\ell} \left(d_i^B - c_i^B\right)^2$
- Parties jointly compute inner product $\sum_{i=1}^{\ell} (\boldsymbol{d_i^A} - \boldsymbol{c_i^A})\,(\boldsymbol{d_i^B} - \boldsymbol{c_i^B})$

SECURE EUCLIDIAN DISTANCE

dist$([d], [c])$
$= (d_1 - c_1)^2 + (d_2 - c_2)^2$



- Parties jointly compute inner product $\sum_{i=1}^{\ell} (d_i^A - c_i^A)\ (d_i^B - c_i^B)$
- Previous work: using generic inner product
- Our observation:
  - Need to compute inner products between 1 point d vs many clusters c
  - Point d_i is fix during all iterations
  - #cluster << #datapoint
- We rewrite

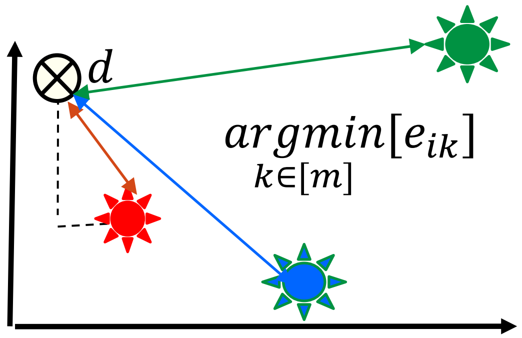
$$(d_i^A - c_i^A)(d_i^B - c_i^B) = d_i^A\ (d_i^B - c_i^B) - d_i^B\ c_i^A + c_i^B\ c_i^A$$

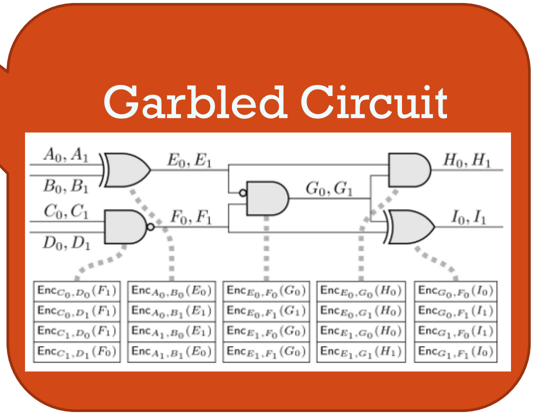
- Applying inner product based on Oblivious Transfer (OT)
  - Reuse OT for d_i^A when computing $d_i^A\ (d_i^B - c_i^B)$
  - Reuse OT for d_i^B when computing $d_i^B\ c_i^A$

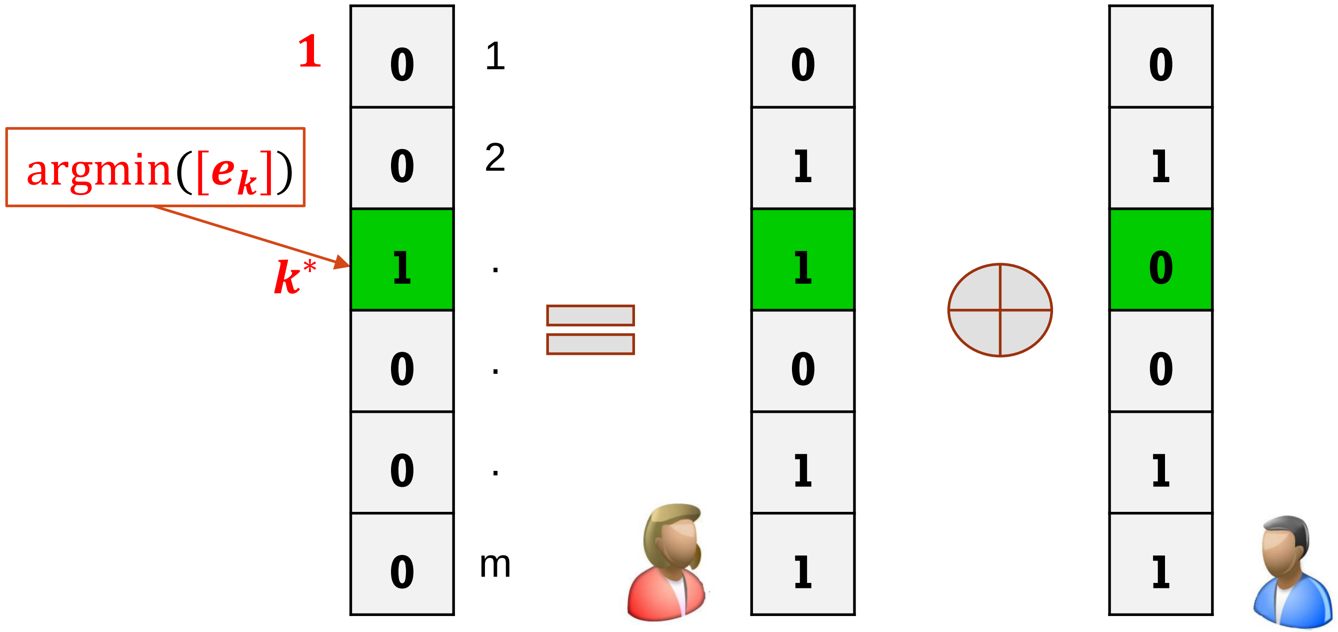
=> 32-148x faster than prior work

ASSIGN DATA TO CLUSTER

- Input: shared Euclidian distances between a point and m clusters $[\boldsymbol{e_k}]$
- Goal:
  - Find closest cluster $[\boldsymbol{k}^*] = \underset{\boldsymbol{k}\in[\boldsymbol{m}]}{\operatorname{argmin}}[\boldsymbol{e_k}]$
  - Assign the point $[d]$ to the closest cluster $[c_{k^*}]$
- Most of prior works reveal $\boldsymbol{k}^*$
- Ours: hide $\boldsymbol{k}^*$
  - Presenting $\boldsymbol{k}^*$ as a binary vector



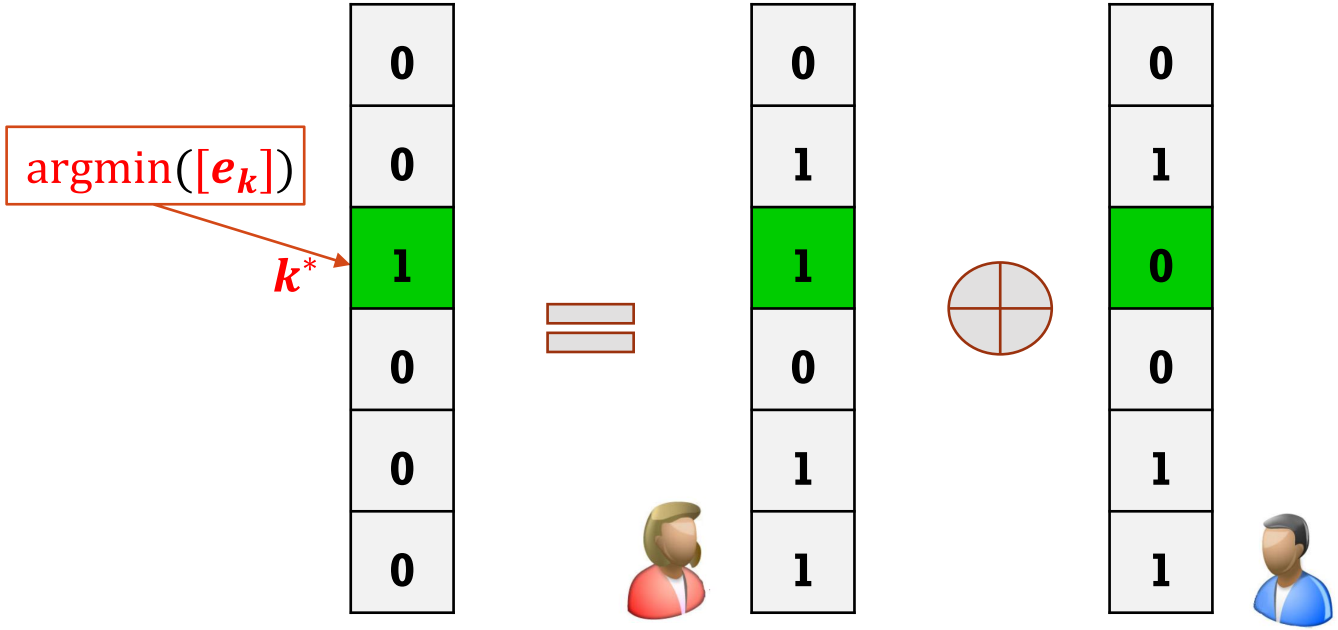




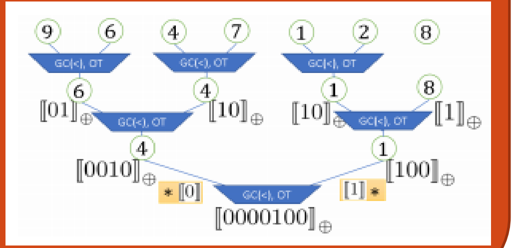
ASSIGN DATA TO CLUSTER

- Input: shared Euclidian distances between a point and m clusters $[\boldsymbol{e}_k]$
- Goal:
  - Find shortest distance $[\boldsymbol{k}^*] = \underset{k \in [m]}{\operatorname{argmin}} [\boldsymbol{e}_k]$
  - Assign the point $[d]$ to the closest cluster $[c_{k^*}]$
- Most of prior works reveal $\boldsymbol{k}^*$
- Ours: hide $\boldsymbol{k}^*$
  - Present ing $\boldsymbol{k}^*$ as a binary vector

$[\boldsymbol{e}_k]$: arithmetic shares
Garble circuit: Yao shares
$\Rightarrow$ convert arithmetic shares to Yao's
$\Rightarrow$ Not cheap!
$\Rightarrow$ We design a better circuit (see paper)

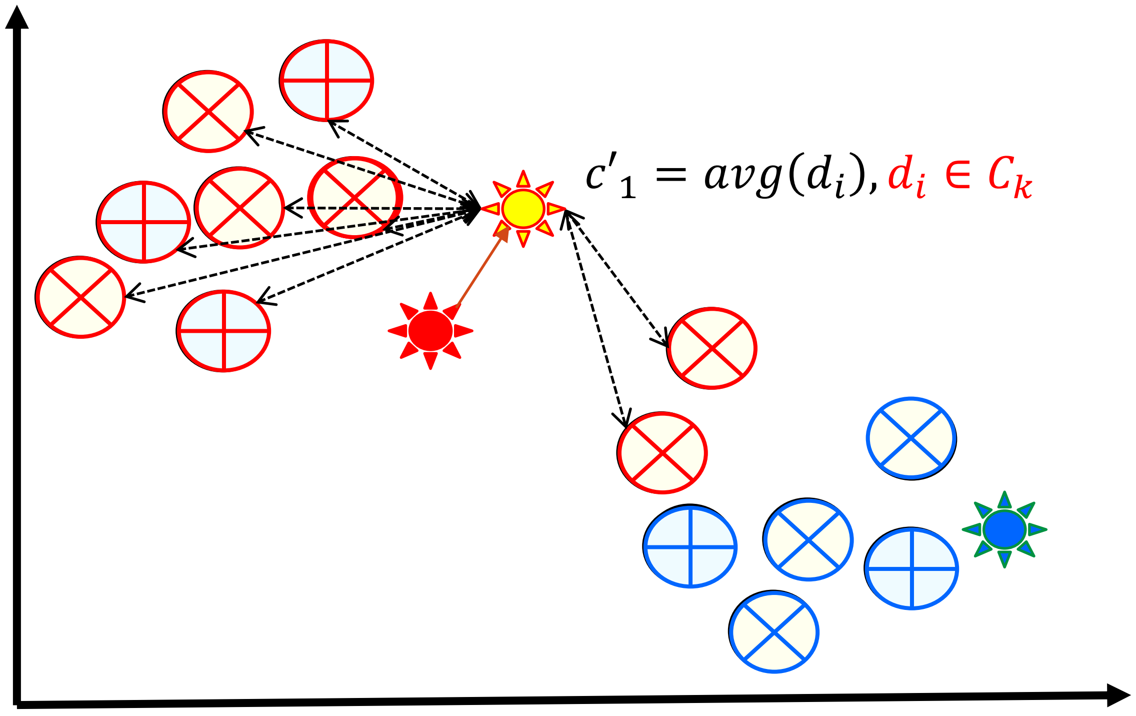


Customized Garbled Circuit

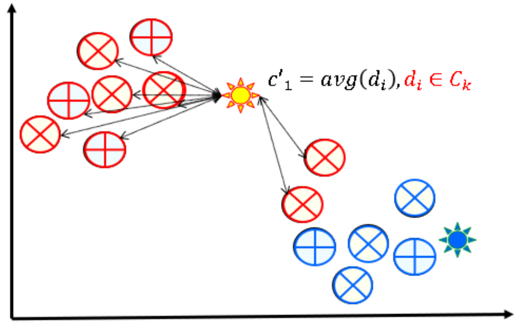


UPDATE CLUSTER

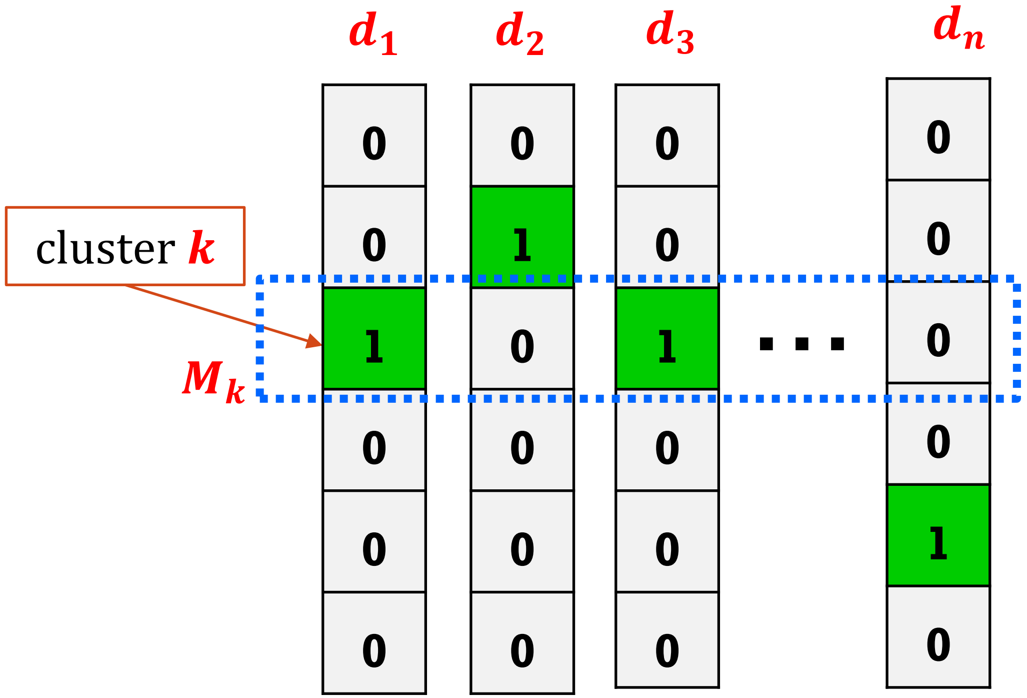
- Recalculate the new cluster center by $\boldsymbol{c_k = avg(d_i), d_i \in C_k}$



UPDATE CLUSTER



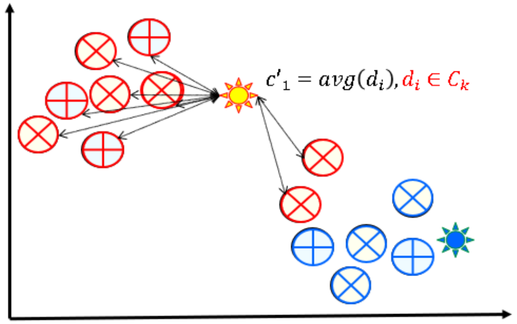
- Recalculate the new cluster center by $c_k = avg(d_i), d_i \in C_k$
- From previous step, we have a bit vector indicated whether $d_i \in C_k$
- Average $avg(d_i)$ can be computed as:



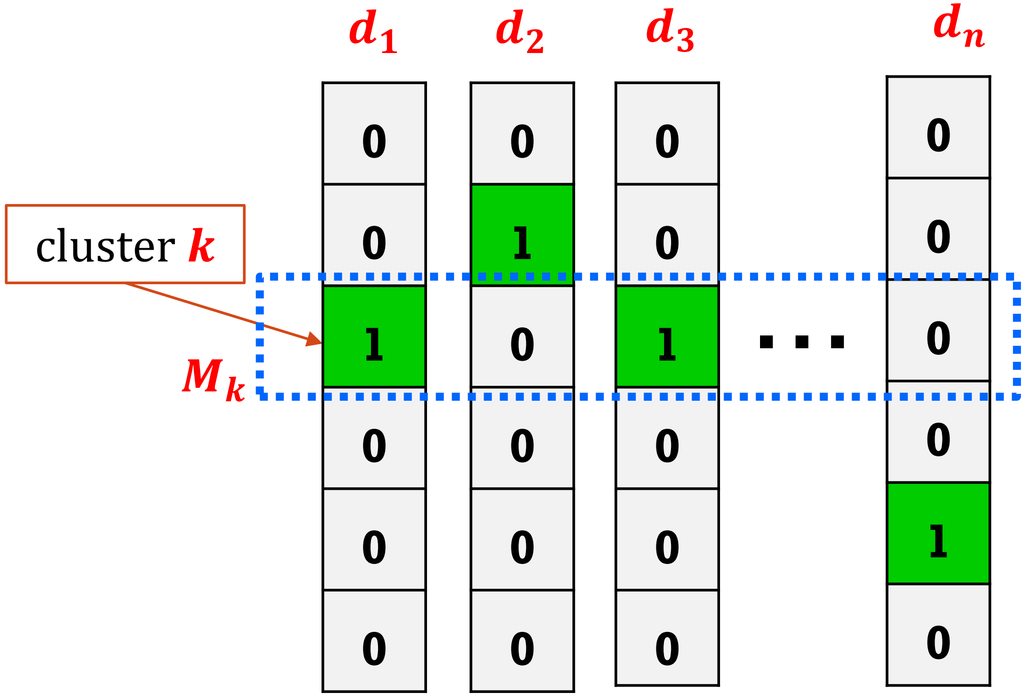
$$c_k = \frac{\sum_{i=1}^{n} M_k[i] * d_i}{\sum_{i=1}^{n} M_k[i]}$$

where M_k and d_i are secretly shared among two parties

UPDATE CLUSTER



- Recalculate the new cluster center by $c_k = avg(d_i), d_i \in C_k$
- From previous step, we have a bit vector indicated whether $d_i \in C_k$
- Average $avg(d_i)$ can be computed as:

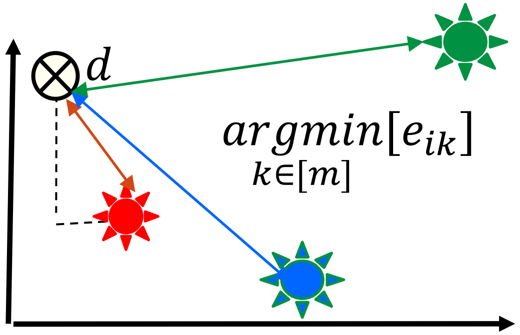


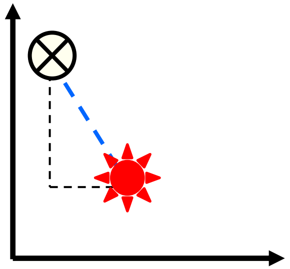
$$c_k = \frac{\sum_{i=1}^{n} M_k[i] * d_i}{\sum_{i=1}^{n} M_k[i]} = \frac{\sum_{i=1}^{n} (M_k^A[i] \oplus M_k^B[i]) * (d_i^A + d_i^B)}{\sum_{i=1}^{n} (M_k^A[i] \oplus M_k^B[i])}$$

where M_k and d_i are secretly shared among two parties

- Direct Solution:
  - Convert Boolean share to arithmetic share
  - Secure multiplication
- Better Solution based Oblivious Transfer (see paper)

PRIVACY-PRESERVING K-MEANS CLUSTERING

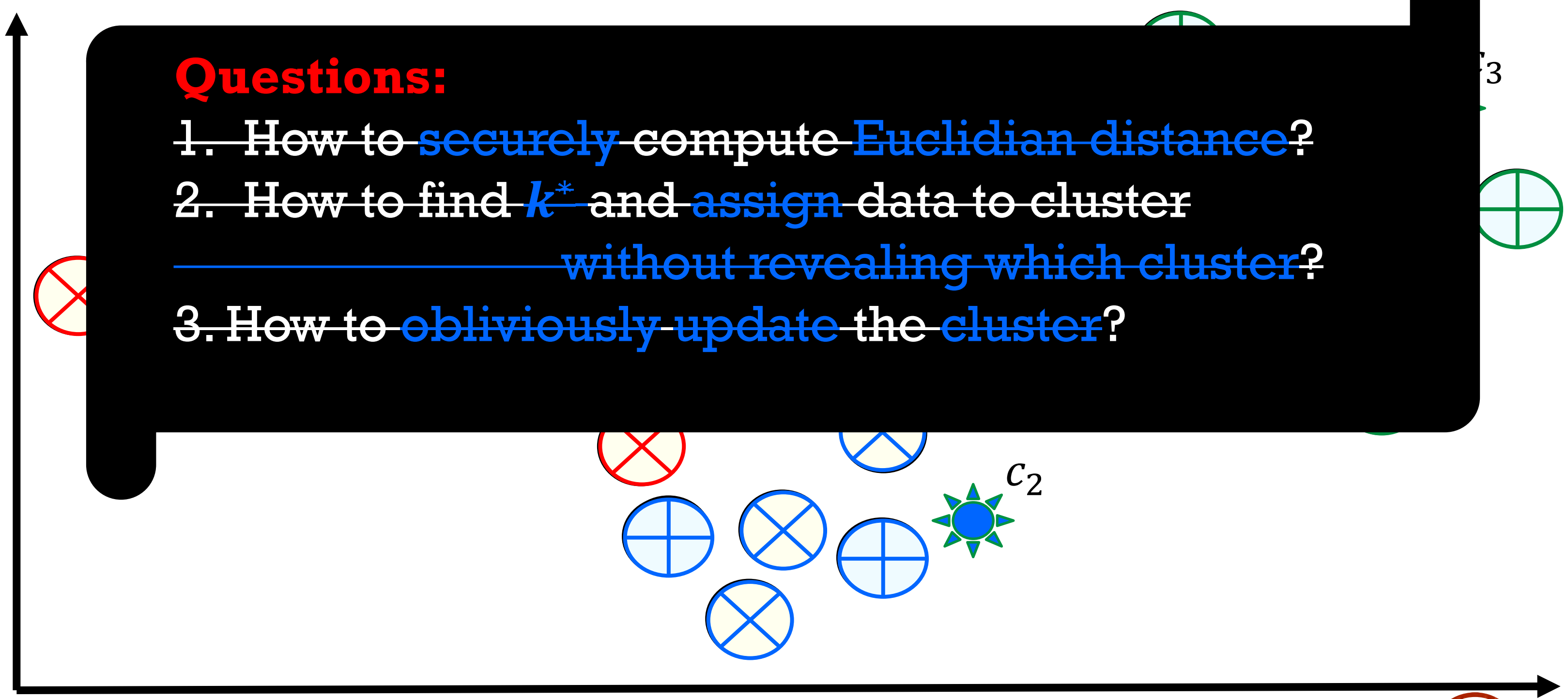




- Inputs:
  - Secret share dataset as $\{[d_1], \ldots, [d_n]\}$
  - #clusters m
- Algorithm:
  - Pick m random clusters $\{[c_1], \ldots, [c_m]\}$
  - Until clustering *converges* (or other stopping criterion):
    - For each data point $\boldsymbol{d_i}$:
      - For each cluster $\boldsymbol{c_k}$:
        - Compute the distance between $\boldsymbol{d_i}$ and $\boldsymbol{c_k}$ as $[e_{ik}]$ =dist($[d_i], [c_k]$)
      - Find closest cluster center

        $[\boldsymbol{k^*}] = \underset{\boldsymbol{k \in [m]}}{\boldsymbol{argmin}}[\boldsymbol{e_{ik}}]$
      - Assign the data point $\boldsymbol{d_i}$ to the closest cluster center $\boldsymbol{k^*}$
    - For each cluster $\boldsymbol{c_k}$:
      - Compute the average of the points assigned to the cluster as

        $[\boldsymbol{c'_k}] = \boldsymbol{avg}([\boldsymbol{d_i}]), \boldsymbol{d_i} \in \boldsymbol{C_k}$
      - Update $\boldsymbol{c_k} = \boldsymbol{c'_k}$



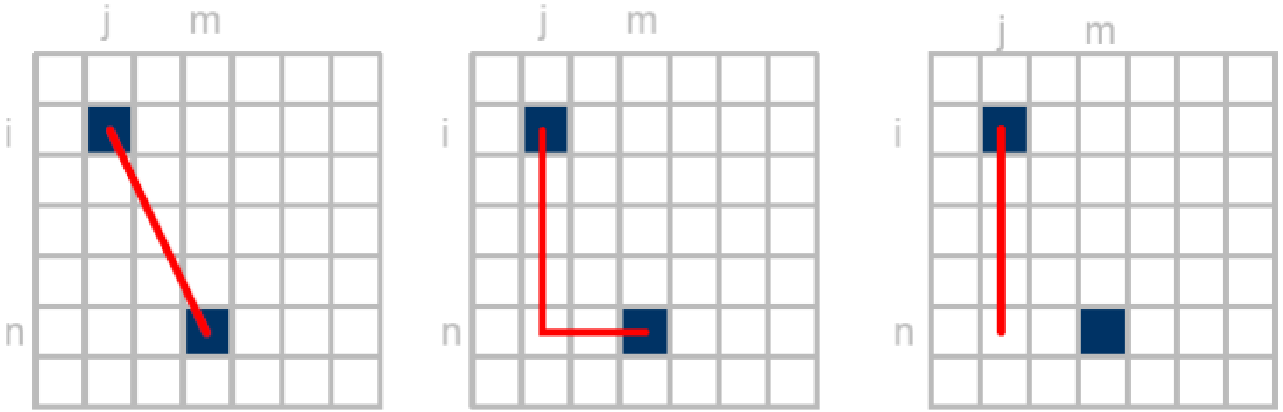
Questions:

~~1. How to securely compute Euclidian distance~~?

~~2. How to find k^* and assign data to cluster without revealing which cluster~~?

~~3. How to obliviously update the cluster~~?

PERFORMANCE: DISTANCE MEASURES



Euclidean	Manhattan	Chessboard
$=\sqrt{(i-n)^2 + (j-m)^2}$	$= \lvert i-n \rvert + \lvert j-m \rvert$	$= \max[\, \lvert i-n \rvert, \lvert j-m \rvert \,]$

- Manhattan metric and Chessboard metric are considered as alternative distance metrics in some ML applications.

- Our **amortized** Secure Euclidean Squared Distance (SESD) cost is 8.9 × − 38.5× faster than the cost of computing a Manhattan distance, and 10.5 × − 40.5× faster than that of Chessboard distance.

Distance Metric		Dimension d 2	3	4	10
Manhattan		1.163	1.623	1.96	4.763
Chessboard		1.222	1.711	2.294	5.791
SESD	$\{K = 4, T = 10\}$	0.094	0.155	0.219	0.474
	$\{K = 4, T = 20\}$	0.079	0.123	0.164	0.398
	$\{K = 16, T = 10\}$	0.036	0.043	0.066	0.172
	$\{K = 16, T = 20\}$	0.031	0.042	0.063	0.163

Running time in millisecond per a distance metric with d dimension. Data size is n = 2^12 , K, T is the number of clusters, and number of iterations, respectively.

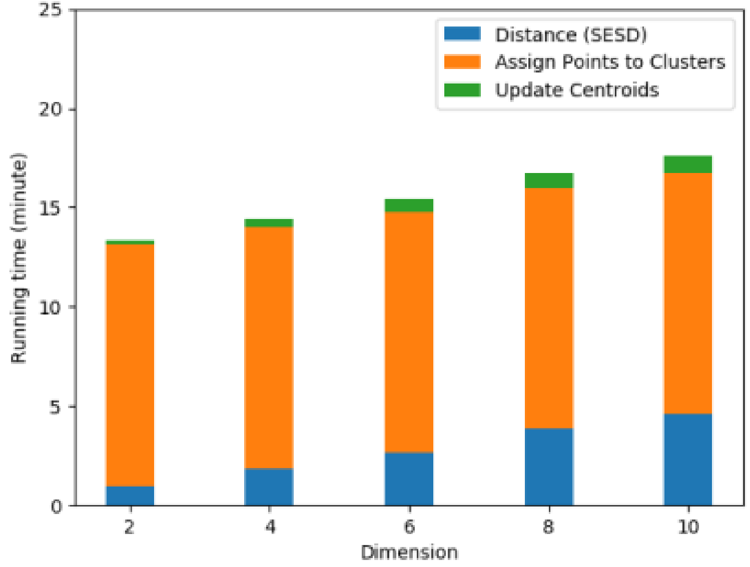
PERFORMANCE

Experiments with Generated Dataset

Parameters			RunTime (minute)				Communication (MB)			
n	K	T	Distance (SESD)	Assign Points to Clusters	Update Centroids	Total	Distance (SESD)	Assign Points to Clusters	Update Centroids	Total
10^4	2	10	0.65	1.14	0.13	1.92	200	2330	10	2540
		20	0.95	2.29	0.26	3.5	398	4660	20	4878
	5	10	0.73	4.61	0.47	5.81	496	8760	40	9296
		20	1.18	9.23	0.94	11.35	989	17520	80	18589
10^5	2	10	5.69	11.12	1.2	18.02	1932	21400	140	23472
		20	10.38	22.25	2.4	35.04	3985	42800	280	47065
	5	10	5.77	47.18	5.13	58.09	4969	85630	340	90939
		20	11.13	94.35	10.27	115.75	9927	171260	680	181867

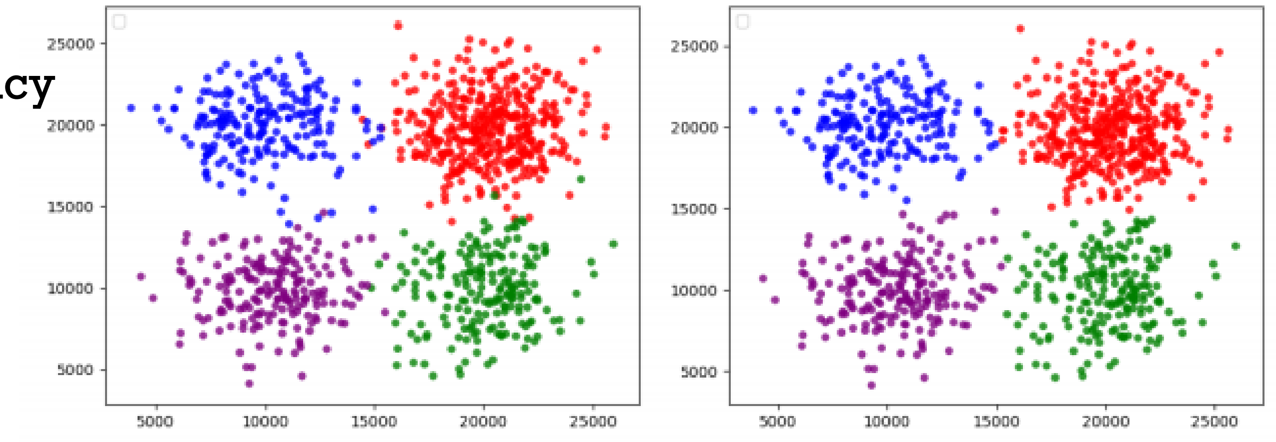
Running time in minute and communication cost of our privacy-preserving clustering protocol, where n, K is the size of database and the number of clusters, respectively, T is number of iterations, dimension d = 2, and bit-length = 3

Experiments with Different Dimensions



Running time (in minute) of our privacy-preserving clustering protocol, where dataset size is 10,000 points, dimension is d ∈ {2, 4, 6, 8, 10}, the number of cluster and iterations are 9 and 15, respectively

Accuracy



(a) Ground Truth Model [2] (b) Plaintext and Privacy-Preserving K-means Model.

Comparison of accuracy for privacy-preserving, plain-text, and ground truth model. Our privacy-preserving model achieves the same accuracy as the plain-text model, which reaches 95% accuracy compared to the expected ideal clusters

THANK YOU

- Paper: https://eprint.iacr.org/2019/1158.pdf
- Code: https://github.com/osu-crypto/secure-kmean-clustering