Aditya Hegde, Helen Möllering, Thomas Schneider, and Hossein Yalame

SoK: Efficient Privacy-preserving Clustering

Abstract: Clustering is a popular unsupervised machine learning technique that groups similar input elements into clusters. It is used in many areas ranging from business analysis to health care. In many of these applications, sensitive information is clustered that should not be leaked. Moreover, nowadays it is often required to combine data from multiple sources to increase the quality of the analysis as well as to outsource complex computation to powerful cloud servers. This calls for efficient privacy-preserving clustering. In this work, we systematically analyze the state-of-the-art in privacy-preserving clustering. We implement and benchmark today's four most efficient fully private clustering protocols by Cheon et al. (SAC'19), Meng et al. (ArXiv'19), Mohassel et al. (PETS'20), and Bozdemir et al. (ASIACCS'21) with respect to communication, computation, and clustering quality. We compare them, assess their limitations for a practical use in real-world applications, and conclude with open challenges.

Keywords: Privacy-preserving Protocols, Clustering, Secure Computation

1 Introduction

In today's world, machine learning (ML) algorithms are widely used to categorize and classify large amounts of data. Applications range from spam filtering over fraud detection, stock market analysis to health diagnostic [1–4]. Moreover, many large IT companies, including Microsoft, Facebook, Google, and Apple, collect massive amounts of data to perform analyses for their commercial benefit [5]. Clustering is a popular unsupervised learning technique and plays a crucial role in data processing and analysis. It divides a set of given input data into subgroups of elements with similar properties.

Aditya Hegde: IIIT-Bangalore, E-mail: aditya.shridhar@iiitb.org (This work was done when the author was intern at the Technical University of Darmstadt)
Helen Möllering, Thomas Schneider, Hossein Yalame: Technical University of Darmstadt, E-mail: lastname@encrypto.cs.tu-darmstadt.de

Cluster analysis is being utilized in various fields with extremely sensitive data such as medical imaging [4] and market research [6], to name a few. Moreover, data protection regulations such as the General Data Protection Regulation (GDPR) in the EU and the Health Insurance Portability and Accountability Act (HIPAA) in the US prohibit companies from sharing sensitive user information. Nevertheless, combining data from different sources, e.g., different hospitals, broadens the database and offers more meaningful, credible, and high-quality clustering results. Additionally, it is often needed to outsource the expensive clustering of large amounts of data to powerful cloud servers. These requirements emphasize the need for privacy-preserving clustering to preserve the privacy of data.

Consequently, a series of efforts have been made to protect the privacy of sensitive input data in clustering through two paradigms for secure computation that can also be combined. The first paradigm leverages homomorphic encryption (HE) [7–9]. HE allows to directly compute functions on encrypted data. The second paradigm uses secure multi-party computation (MPC) [10, 11]. MPC allows mutually distrusting parties to collaboratively compute a joint function over their respective private data. However, these works only cover a few clustering algorithms so far: K-means, K-medoid, Mean-shift, Gaussian Mixture Models Clustering (GMM), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), hierarchical clustering (HC), Affinity Propagation, and Mean-shift. Moreover, we found that only ten works (cf. Tab. 1) provide full privacy protection according to the ideal functionality for privacy-preserving clustering, i.e., they leak nothing beyond the output (cf. §3.1).

Even revealing little and at the first glance minor information during the clustering can have severe consequences for the data privacy of individuals. For example, when using clustering for the segmentation of medical images [4] between two hospitals, revealing the cluster sizes and assignments in each clustering iteration leaks information about how many patients with similar characteristics are input by the other party even before the clustering stabilizes and a final result is reached. Unintended common characteristics between patients might be leaked even though they are only temporarily assigned to one cluster (due to these characteristics which

Algorithm	Paper	PETs	Scenario	Data	Output	Efficiency
K-means	[12, CCS'07]	HE+ASS	2PC	a	final centroids	✗†
	[13, CIC'15]	HE	Outsourcing, 2 Servers	h	final centroids	✗†
	[14, SAC'18]	HE	Outsourcing, 1 Server	–	final centroids	✗⋆
	[15, CLOUD'18]	HE	Outsourcing, 2 Servers	–	cluster sizes	✗†
	MPC-KMeans [11, PETS'20]	GC	Outsourcing, 2 Servers *or* 2PC	h	final centroids	✓
Mean-shift	HE-Meanshift [9, SAC'19]	HE	Outsourcing, 1 Server	–	final centroids	✓
Affinity Propagation	[16, SECRYPT'21]	ASS	Outsourcing or MPC	a	final clusters	✗‡
DBSCAN	[17, S&P'13]	GC	2PC	h	cluster labels, centroids/size possible	✗¶
	ppDBSCAN [18, ASIACCS'21]	GC+ASS	Outsourcing, 2 Servers *or* 2PC	a	cluster labels, centroids/size possible	✓
Hierachical Clustering	PCA/OPT [19, ArXiv'19]	HE+GC	2PC	h	final dendogram	✓

† Computationally expensive due to use of Paillier's HE and no parallelization.
⋆ Costly computation due to use of bit-wise encryption. MPC-KMeans [11] outperforms this scheme by 5000× for 400 data records.
‡ [18] is 194× faster than this scheme for 400 data records.
¶ [18] is 5× faster than this scheme for a dataset size of 500 data records.

Table 1. Fully privacy-preserving clustering protocols (cf. §3.1). HE is homomorphic encryption [7], ASS is arithmetic secret sharing [20], and GC is garbled circuits [21]. v indicates vertically partitioned data, i.e., the data owners hold the values for a subset of parameters from all data records. h indicates horizontally partitioned data, where the data owners hold complete data records with all parameters, a is arbitrarily partitioned data, and " – " indicates the scheme has only one data owner. Schemes that were implemented and benchmarked in §4 are highlighted in gray.

would not have been revealed in the final result). An even more severe privacy breach is demonstrated in [22] where leaking the results of comparison of distances between data records and a threshold can enable to accurately approximate the original data record held by another party. With this, complete patient records could be extracted when clustering medical data. To summarize, it is difficult to concretely determine the effects of leaking intermediate information in advance for all possible constellations. Hence, privacy research should focus on designing efficient private clustering protocols that do not leak anything beyond what can be inferred from the output, i.e., provide full privacy.

Related Work. Privacy-preserving machine learning (PPML) is a hot topic in recent privacy research [23–26]. To provide a better overview over the exploding research field, several surveys have been done. Haralampieva et al. [27] survey existing frameworks in the context of private image classification. An overview about frameworks for private neural network inference is given in [28]. Protocols used for private machine learning training are investigated in [29]. Similarily, Tanuwidjaja et al. [30] summarize existing works on privacy-preserving deep learning and issues when using these schemes as well as possible attacks on private deep learning. Kiss et al. [31] systematically review the state-of-the-art approaches to private decision tree evaluation.

All previous surveys focus on privacy-preserving supervised learning where a training dataset with labelled samples (i.e., known input-output pairs) is used to train a model that can later be used to classify new data records. In contrast, our survey focuses on clustering, a popular *unsupervised* machine learning (ML) technique, which detects unknown patterns in unlabelled data so no "training" of a model is needed. In our work, we systematically survey and evaluate the state-of-the-art in private clustering using secure computation techniques.

An orthogonal line of research uses differential privacy (DP) to protect privacy-preserving machine learning (PPML), including clustering [32–37], against information leakage. Abadi et al. [38] and Shokri et al. [39] provide comprehensive surveys on differentially private deep learning. Generally, the noise added to achieve DP reduces utility whereas secure computation has higher complexity. Hence, DP-based and secure computation-based protocols are not directly comparable and we leave a survey on DP-based clustering for future work.

Our Contributions and Outline. After presenting the preliminaries of privacy-preserving clustering in §2, our Systematization of Knowledge (SoK) paper provides the following core contributions:

– The first comprehensive review and analysis of existing techniques and protocols used for privacy-preserving clustering with respect to security models, privacy limitations, efficiency, and further aspects. We also provide guidelines on how to choose an appropriate privacy-preserving clustering scheme for a specific application (§3).
– An empirical evaluation of the four most efficient and fully private clustering schemes [9, 11, 18, 19], cf. Tab. 1, on a range of criteria, including clustering quality, security and privacy, and runtime/communication overhead (§4). Based on these insights, we provide an analysis of the practicality of the four protocols for real-world applications based on our results from the benchmarking (§5).

– An implementation of the clustering protocol of [9] and [19] in C++17. Implementations of the remaining two protocols that we also evaluate [11, 18] are publicly available. Our code is available at https://encrypto.de/code/SoK_ppClustering.

	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
1) Cluster Shapes	−	−	○	+	+	−	−	−
2) Large Datasets	○	−	−	−	−	−	+	○
3) Update Input Data	+	−	−	○	+	+	+	+
4) Nominal Variables	−	+	+	−	+	+	−	−
5) Outliers	−	+	○	−	+	○	+	○
6) Input Order	+	+	+	+	○	+	−	+
7) Storage	+	−	−	+	+	−	+	+
8) # Parameters	−	○	−	○	○	−	○	−
Full privacy	✓	✓	✓	✓	✓	✗	✗	✗

Table 2. Comparison of clustering algorithms with respect to the aspects explained in §2.1: (a) K-means, (b) Affinity Propagation, (c) Single/Complete Linkage HC, (d) Mean-shift, (e) DBSCAN, (f) K-medoid, (g) BIRCH, and (h) GMM. + denotes that the clustering algorithm performs well with respect to the indicated aspect, ○ denotes an average performance, and − indicates that it has some weaknesses. ✓ indicates that a fully privacy-preserving clustering protocol is available and ✗ that it is not available yet.

2 Preliminaries

2.1 Clustering

Clustering is a well-known unsupervised machine learning (ML) technique, i.e., it deals with detecting unknown patterns in unlabeled data. Concretely, it groups similar input records (*internal homogeneity*) in *clusters* while records belonging to different clusters should be maximally different (*external separation*) [40–42].

Clustering consists of four components: feature selection/normalization, a proximity measure to determine similarity/dissimilarity, the clustering algorithm, and the output assessment [41, 42]. However, most prior works on privacy-preserving clustering mainly focus on a specific clustering algorithm. For example, the proximity measure is typically chosen to enable efficient computation using cryptographic techniques [14, 19]. Furthermore, mostly continuous values are considered while clustering can generally be applied to any kind of variable (i.e., also discrete or nominal values) [42].

Clustering algorithms can be split in two classes: hard and soft (fuzzy) clustering. In hard clustering, each input data record is assigned to exactly one cluster. In soft clustering, data records can be assigned to several clusters with a certain probability. All works on privacy-preserving clustering that we investigated in this work except from [43] have only tackled hard clustering.

Properties of Good Clustering. Records are assigned to the same cluster given they are *similar*. However, (dis)similarity heavily depends on the chosen proximity measure. Additionally, clustering algorithms were designed having specific problems in mind such that they exhibit biases that affect their performance when the assumed conditions are not fulfilled. Therefore, according to Xu and Wunsch [41], no clustering algorithm is universally superior and a good clustering algorithm should be able to cope with: 1) arbitrarily shaped clusters, 2) large datasets, 3) updates with new records without having to cluster old records again, 4) numerical (i.e., discrete and continuous) and nominal variables, and 5) outliers. Furthermore, it should: 6) be insensitive to the order of input records, 7) provide acceptable storage requirements, and 8) minimize the number of input parameters. Finally, it should also be able to handle 9) high-dimensional data records.

Clustering Algorithms

In the context of privacy-preserving clustering, four different types of clustering have been studied so far: partitioning-based [8, 11, 14, 16], distribution-based [44, 45], density-based [18, 46, 47], and hierarchical clustering [19, 48–50]. In the following, we summarize these four clustering types and compare the respective algorithms w.r.t. the properties listed before in §2.1. Due to space limitations, we only provide the details of this evaluation for the three algorithms [51–53] for which fully private protocols were proposed (cf. §3.2) and that we benchmark in §4. Details of the other algorithms are given in Appx. A.

Partitioning-based Clustering. Partitioning-based clustering splits the input into K non-overlapping clusters. Typically, an initial random partition is iteratively improved given an objective function [54].

A well-known example is K-means [51]. It has a computational complexity of $\mathcal{O}(NKt)$ and a space complexity of $O(N)$ [40] for dataset size N, K clusters, and t clustering iterations. Furthermore, K-means can only cluster convexly-shaped clusters, cannot to appropriately handle outliers, and requires to pre-determine the number of clusters K [40]. If the initial partitioning, i.e., the centroid initialization, is done at random, K-means is not deterministic. It may converge to a local optimum [55]. The input order does not affect the clustering result. As the centroids are determined by averaging, K-means is not suitable for nominal variables [56]. New data records typically require only a few additional clustering iterations because they normally do not significantly change the result. Other partitioning-based clustering algorithms that were investigated in the context of privacy-preserving clustering are the closely related K-medoids [57], Kernel K-means [58], Possibilistic C-means [43], as well as Affinity Propagation [16].

Hierarchical Clustering. Hierarchical Clustering (HC) algorithms can be classified into agglomerative and divisive approaches. In agglomerative algorithms, each data record forms an own cluster in the beginning and the clusters are then iteratively merged together based on their proximity. Divisive algorithms follow the opposite approach and start with all elements in one cluster which is then iteratively split up [52, p. 71-72]. HC algorithms output a binary tree/dendrogram[1] where each leaf represents a record and nodes indicate a merge of two similar clusters into one. The root combines all records into a single cluster [40].

As divisive HC exhibits an immense overhead for examining the optimal splits ($2^{N-1}-1$ possibilities [40], where N is the dataset size), mostly agglomerative algorithms have been observed in practice. Traditionally, three merging methods were used: (1) single, (2) complete, and (3) average linkage. Single linkage merges the two clusters with the closest two elements, complete linkage merges the two clusters whose maximally distant pair of elements are closest among all pairs of clusters, and average linkage merges the two clusters that have the smallest average of all pairwise distances of their elements [52, p. 76-77] [59].

Naive HC has computation complexity $\mathcal{O}(N^3)$ and space complexity $\mathcal{O}(N^2)$ [42]. Some HC-based algorithms (e.g., single linkage) cannot detect some cluster shapes. They do not incorporate a notion of noise, but are relatively insensentive to outliers. HC requires to pre-determine the number of clusters K that are obtained by cutting the tree at the respective level [40]. HC needs a restructuring of the tree if new data records are added after the first clustering. Nevertheless, HC can handle any type of variable and the input order does not affect the result.

Density-based Clustering. These algorithms use a density-based neighborhood notion such that input records that lay together in a dense area form a cluster. Examples are Mean-shift [53] and DBSCAN [60]. Mean-shift has time complexity $\mathcal{O}(N^2 t)$, where N is the dataset size and t is the number of iterations, which makes it inefficient for large datasets. It can handle any cluster shape and flexibly determine the number of clusters K based on the input data. Additionally, the input order does not affect the results. However, the value of the bandwidth h in the Kernel Density Estimator (KDE) used in Mean-shift can significantly affect its performance. A too large h merges distinct clusters while a too small h splits one cluster into multiple smaller groups. The performance also deteriorates for high dimensional data due to the "curse of dimensionality" in the KDE. Similarly, noisy features can hamper the performance [61]. Mean-shift does not incorporate a notion of noise. An update with new records can change the KDE and the local maximas, thus requiring a rerun of the entire algorithm. However, in practice, the new points can be assigned to the cluster containing the nearest mode if the change in the KDE is not significant.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN [62]) specifically recognizes noisy elements and marks them as outliers. It detects arbitrarily shaped clusters and flexibly determines the number of clusters in a dataset based on two input parameters, namely the minimal cluster size and the maximal distance between two clusters. Especially the second parameter can be difficult to determine and DBSCAN cannot correctly handle clusters with significantly different densities [63]. Generally, if appropriate distance measures are chosen, any type of parameter can be clustered. Moreover, the input order does only affect the clustering result in exceptional cases where border elements lay in the range of more than one cluster. If additional data records shall be clustered after a first clustering was finished, their neighbors have to be determined to assess if they can be added to previously created clusters. Otherwise, they may create a new cluster with other outliers, but no completely new clustering is needed. Naive DBSCAN needs $\mathcal{O}(N)$ memory and has computation complexity $\mathcal{O}(N^2)$ [62, 63].

Distribution-based Clustering. Distribution-based clustering algorithms assume that clusters are drawn from an unknown mixture of distributions and aim at approximating the original distributions (i.e., the type and parameters) as well as the number of different distributions (i.e., the number of clusters) [41, 42]. A well-known example for distribution-based clustering algorithms are Gaussian Mixture Models (GMM) using the Expectation-Maximization (EM) algorithm [64].

1 A dendogram is a graph representing a tree structure.

Comparison of Clustering Algorithms. Modifications proposed for the clustering algorithms to fix some weaknesses of the original often introduce other problems. Therefore, it is difficult to evaluate them with respect to the general requirements for clustering algorithms (cf. §2.1).

In Tab. 2, we compare the eight baseline clustering algorithms for which privacy-preserving protocols have been proposed with respect to the properties of good clustering algorithms listed in §2.1. We did not include the effect of property 9), i.e., high dimensionality, because it is often not directly linked to the clustering algorithm. Instead, a large number of variables often requires using feature reduction techniques.

2.2 Cryptographic Building Blocks

In the following, we summarize secure computation techniques and respective security models.

Secure Computation. There are two main paradigms for secure computation: Homomorphic encryption (HE) and multi-party computation (MPC). HE [7, 65, 66] enables operations on a set of ciphertexts such that the resulting ciphertext contains the result of a function on the corresponding plaintexts. MPC allows two or more mutually distrusting parties to jointly compute a function on their private inputs. Two well-known generic approaches for MPC are based on garbled circuits (GC) [21] and secret-sharing (SS) [20, 67]. As an example for a SS-based technique, the GMW protocol [20] represents a function as Boolean/Arithmetic circuit and the values are secret-shared using XOR or Arithmetic secret sharing (ASS). Another type of SS is Shamir's secret sharing (SSS) [67].

Security Models. Two main security models have been considered in privacy-preserving clustering: In the *semi-honest/passive security model*, the adversary [68] is assumed to honestly follow the protocol, but tries to learn additional information about the private inputs of other parties. Though this model is weaker than the *malicious* model, that even protects against deviations from the protocol specification, it facilitates practically-efficient applications especially for privacy-preserving machine learning (PPML) [69]. *Full threshold* security means that up to $N-1$ parties can collude without jeopardising privacy while *honest majority* security requires the majority of the parties to not collude.

3 Privacy-preserving Clustering

In this section, we first define privacy-preserving clustering. Then, we categorize and analyse the existing privacy-preserving clustering protocols to conclude which protocols offer good efficiency with strong privacy guarantees. Afterwards, we discuss possible applications and provide indications on how to choose appropriate privacy-preserving clustering schemes for these.

3.1 Functionality and Requirements

In an ideal world with a trusted third party (TTP), all involved parties send their input data to the TTP. The TTP then performs the clustering and returns the output to the parties. The output can vary depending on the application requirements and clustering algorithm. For example, the output can be the cluster centroids or it can be the cluster label for each data record.

We identified the following requirements for privacy-preserving clustering:

Privacy. According to the *ideal functionality* a privacy-preserving clustering protocol must not leak information other than what can be derived from the output of the protocol to be considered as *fully privacy-preserving*. Importantly, this includes that all operations must be obliviously realized and all intermediate results must be kept private.

Efficiency. A privacy-preserving clustering scheme must be efficient in terms of communication and runtime. This means that it must scale well with respect to the dataset size N, the number of clusters K, and the dimensionality d of the input records.

Clustering Quality: A privacy-preserving clustering scheme must offer a good clustering quality of the results independent of a dataset's properties. Specifically, the requirements of good clustering listed in §2.1 should be fulfilled.

Flexibility. A privacy-preserving clustering scheme should ideally be flexibly usable for *outsourcing* [70] and *multi-party computation*. In an outsourcing scenario, one or multiple data owners outsource their data and the computation to untrusted non-colluding parties [70]. Here, the data owners can even be malicious (cf. §2.2). In multi-party computation, several parties interactively compute the clustering on their joint dataset.

3.2 Existing Private Clustering Protocols

In this subsection, we categorize the existing works on privacy-preserving clustering with respect to the underlying plaintext clustering algorithm, security model, scenarios for which protocols where designed, data distributions, used secure computation techniques as well as privacy and efficiency (cf. §3.1). We discuss the strengths and weaknesses of these schemes with respect to these criteria. Tab. 3 contains on overview of all 59 works on privacy-preserving clustering with secure computation techniques that we are currently aware of. It indicates the respective security model, used secure computation techniques, common types of leakages of intermediate values, the type of output, which and how many parties are involved in the protocol, the data partition, and other issues.

Plaintext Clustering Algorithms. Eight clustering algorithms have been investigated in the context of privacy-preserving clustering: K-means (including the two variants Kernel K-means [58] and Possibilistic C-means [43]), K-medoids [57, 71], GMM [44, 45], Mean-shift [9], DBSCAN [22, 46, 47, 72–76], baseline agglomerative HC (e.g., single linkage or complete linkage) [19, 48–50, 77, 78], BIRCH [79, 80], and Affinity Propagation [16, 81]. The vast majority of works focuses on the simple K-means algorithm [8, 11–15, 82–106], which enables an efficient parallelization of computation through packing with homomorphic encryption [8, 88, 95] or amortization through batched oblivious transfers [11]. However, as discussed in §2.1, K-means can be used only for very specific applications where the number of clusters is known in advance and the clusters are convexly shaped. We gave an overview of the strengths and weaknesses of these plaintext clustering algorithms in Tab. 2. Generally, the choice of the plaintext clustering algorithm heavily affects the quality of the clustering result. Some works on privacy-preserving clustering exactly reproduce the original algorithms and hence achieve the same accuracy, e.g., [19, 44, 45, 82]. Others deviate from the original algorithms such as when updating the centroids in K-means due to, e.g., normalization/quantization/specific encodings of the plaintext space [8, 9, 14, 22, 88, 95], adaptations of the original algorithm [14], or approximations [14, 43] which either enhance efficiency or are needed because of the underlying secure computation techniques.

Security Models. All works except for [16, 96, 105] consider only the semi-honest security model (cf. §2.2). A few even do not explicitly define their security model [43, 48, 57, 58, 71, 100, 108]. The semi-honest security model assumes that the adversary correctly follows the protocol while trying to gain additional information. However, this strong assumption is not always realistic. Concretely, the use of protocols that are secure against semi-honest adversaries is only acceptable in specific applications where the participants already generally trust each other but are legally not allowed to share data, e.g., hospitals conducting medical analysis or central banks for financial analytics on country-level. We discuss the requirements and implications of applications on the choice of a privacy-preserving clustering scheme in more detail in §3.3.

Scenarios. Generally, privacy-preserving clustering protocols have been designed for two scenarios: Firstly, multi-party computation (MPC, [16, 44–46, 50, 57, 71, 72, 74–76, 82, 85, 87, 89, 91, 94, 96, 99–101, 103]) with the special case of two-party computation (2PC, [11–13, 19, 22, 48, 49, 73, 74, 77–80, 83, 84, 86, 93]), where two or more data owners jointly perform a secure computation protocol ideally such that nothing beyond the output is leaked to each other (cf. §3.1). Some of these protocols [50, 75, 76, 88, 95, 100, 103] also involve one or more additional (semi-trusted) entities, e.g., represented by servers, that assist in the computation. In contrast, other protocols were designed for the outsourcing scenario where one or more data owners outsource computation (and storage) to external parties who ideally perform the clustering for them without learning anything about the input data [8, 9, 14, 15, 43, 47, 58, 90, 92, 97, 98, 102, 104–106, 108]. As outsourcing aims at using external resources, data owners should not be involved in the execution of the protocol and can go offline, but this is often not fulfilled, e.g., in [43, 47, 97, 98, 104, 105, 108]. Some MPC/2PC protocols can also be used for an outsourcing scenario where the data owners secret share their data among multiple non-colluding parties who then perform the clustering [8, 11, 105]. However, whether a 2PC/MPC clustering protocol is usable for outsourcing heavily depends on its design. This is hindered if data owners are actively involved by computing on plaintext input data, e.g., [22, 44–46, 48, 49, 72–75, 82, 88], or a data owner needs to perform intermediate decryptions, e.g., [86, 91, 95].

Data Partition. The data to be clustered in a private manner can be partitioned in three ways when provided by multiple parties. It is horizontally partitioned when each data owner holds complete (but different) data records [9, 13, 44, 45, 47, 49, 50, 73, 75, 77, 78, 84–86, 91, 92, 94–96, 98, 99, 101–103, 105, 106, 108]. The data is vertically partitioned when data owners hold mutually different parameters of the same data records [44,

Algorithm	Scheme	Privacy	Security	PETs	L1	L2	L3	L4	O1	O2	O3	Interactivity (Scenario)	Data	Other issues
K-means	[82, KDD'03]	✗	◐	HE+blinding	(✗)[1]	✗	✗	✗	✗	✓	✗	all data owners (≥ 3)	v	
	[83, KDD'05]	✗	◐	HE+ASS+GC	✓	✓	✗	✗	✓	✓	✗	2PC	a	wrong division
	[84, ESORICS'05]	✗	◐	HE *or* OPE	✗	✓	✓	✗	✗	✓	✗	2PC	h	
	[12, CCS'07]	✓	◐	HE+ASS	✓	✓	✓	✗	✗	✓	✗	2PC	a	
	[85, SECRYPT'07]	✗	◐	blinding	✗	✓	✗	✗	✓	✓	✗	all data owners	v/h	
	[86, AINAW'07]	✗	◐	HE+ASS+OPE	✓	✗	✗	✗	✓	✓	✗	2PC	h	
	[87, PAIS'08]	✗	◐	ASS	✓	✓	✗	✗	✓	✓	✗	all data owners (≥ 4)	v	
	[88, WIFS'09]	✗	◐	HE	✗	✓	✗	✓	✓	✗	✗	data owners + 1 server	h	
	[89, KAIS'10]	✗	◐	HE+ASS	✓	✓	✗	✗	✓	✗	✗	all data owners	h	
	[90, PAISI'10]	✗	◐	SS	✓	✗	✗	✗	✓	✓	✗	Outsourcing ≥ 3 servers	a	
	[91, ISPA'10]	✗	◐	HE	✓	✓	✗	✗	✗	✓	✗	all data owners	v/h	
	[92, WIFS'11]	✗	◐	HE+GC	✓	✗	✓	✓	✓	✗	✗	Outsourcing, 3 servers	h	
	[93, ISI'11]	✗	◐	HE+ASS	(✗)[1]	✗	✗	✗	✓	✗	✗	2PC	v	
	[94, TM'12]	✗	◐	SSS	✗	✗	✓	✗	✗	✓	✗	all data owners	h	distance calculation unclear
	[95, JIS'13]	✗	◐	HE	✗	✓	✓	✗	✓	✗	✗	data owners + 2 servers	h	
	[96, ICDCIT'13]	✗	●	SSS+ZKP	✗	✗	✓	✗	✗	✓	✗	all data owners	h	
	[97, ASIACCS'14]	✗	◐	HE	✗	✗	✗	✗	✓	✓	✗	outsourcing, 1 data owner + 1 server	–	insecure HE [107]
	[98, MSN'15]	✗	◐	HE	✗	✗	✗	✓	✗	✗	✗	outsourcing, data owners + 1 server	h	insecure HE [107]
	[99, IJNS'15]	✗	◐	HE	✗	✗	✗	✗	✗	✓	✗	all data owners	h	
	[13, CIC'15]	✓	◐	HE	✓	✓	✓	✗	✗	✓	✗	Outsourcing, 2 servers	h	
	[100, ICACCI'16]	✗	N/A	SS	✗	✗	✗	✗	✓	✗	✗	arbitrary number of servers	a	
	[101, ISPA'16]	✗	◐	blinding	✗	✗	✗	✓	✗	✓	✗	all data owners (≥ 3)	h	
	[102, SecComm'17]	✗	◐	HE	✓	✗	✗	✓	✗	✓	✗	outsourcing, ≥ 4 servers	h	
	[103, TII'17]	✗	◐	HE	✗	✗	✗	✗	✗	✗	✗	data owners + 1 server	h	
	[14, SAC'18]	✓	◐	HE	✓	✓	✓	✓	✗	✓	✗	Outsourcing, 1 server	–	
	[15, CLOUD'18]	✓	◐	HE	✓	✓	✓	✗	✗	✓	✗	Outsourcing, 2 servers	–	distance calculation unclear
	[108, CCPE'19]	✗	N/A	HE	✗	✗	✗	✗	✗	✓	✗	Outsourcing, 2 data owners + 1 server	h	insecure HE [107]
	[104, TCC'19]	✗	◐	HE	✓	✗	✗	✓	✓	✗	✗	Outsourcing, 1 data owner + $\geq$ 1server(s)	–	
	[105, Inf. Sci.'20]	✗	◐ (●)[2]	HE+GC	✗	✗	✗	✗	✗	✓	✗	Outsourcing, 2 data owners + 1 server	h	
	[106, SCN'20]	✗	◐	HE+SKC	✓	✗	✗	✓	✗	✓	✗	Outsourcing, 3 servers	h	
	[11, PETS'20]	✓	◐	GC	✓	✓	✓	✗	✗	✓	✗	2PC/Outsourcing	h	
	[8, TKDE'20]	✗	◐	HE	✓	✗[3]	✓	✗	✗	✓	✗	Outsourcing, 2 servers	a	
Kernel K-means	[58, KAIS'16]	✗	N/A	PKC	✓	✗	✗	✗	✓	✗	✗	Outsourcing, 1 server	–	security model
Possibilistic C-means	[43, TBD'17]	✗	N/A	HE	✗	✗	✗	✗	✓	✓	✗	Outsourcing, 1 data owner + 1 server	–	
K-medoids	[57, SMC'07]	✗	N/A	HE+blinding	✓	✗	✗	✓	✗	✗	✗	all data owners	v	exhaustive search
	[71, CCSEIT'12]	✗	N/A	HE+blinding	✓	✗	✗	✓	✗	✗	✗	all data owners	v	exhaustive search
GMM	[45, KAIS'05]	✗	◐	blinding	✓	✓	✗	✗	✓	✗	✗	all data owners	h	
	[44, DCAI'19]	✗	◐	ASS	✓	✓	✗	✗	✓	✗	✗	all data owners (> 2)	v/h	
Affinity Propagation	[81, INCoS'12]	✗	◐	HE + blinding	✓	✓	✗	✓	✓	✗	✗	all data owners	v	
	[16, SECRYPT'21]	✓	◐/●	ASS+GC	✓	✓	✓	✓	✓	✗	✗	all data owners/Outsourcing	a	
Mean-shift	**[9, SAC'19]**	✓	◐	HE	✓	✓	✓	✓	✓	✗	✗	Outsourcing, 1 server	–	
DBSCAN	[72, ISI'06]	✗	◐	blinding	✓	✓	✗	✓	✗	✗	✗	all data owners	v	lack of complete protocol
	[73, ADMA'07]	✗	◐	HE+blinding	✓	✗	✗	✓	✓	✗	✗	2PC	v/h	
	[74, IJSIA'07]	✗	◐	PKC+blinding	✓	✓	✗	✓	✓	✗	✗	all data owners	v	
	[75, ITME'08]	✗	◐	HE+blinding	✓	✗	✗	✓	✓	✗	✗	data owners + 1 server	h	
	[22, TDP'13]	✗	◐	HE+blinding	✓	✗	✗	✓	✓	✗	✗	2PC	a	
	[17, S&P'12]	✓	◐/●[5]	GC	✓	✓	✓	✓	✓	✓	✗	2PC	h	
	[46, SIBCON'17]	✗	◐	HE+PKC	✓	✓	✗	✓	✓	✗	✗	all data owners	v	cluster expansion missing
	[47, PRDC'17]	✗	◐	HE	✓	✗	✗	✓	✗	✗	✗	outsourcing, all data owners + 1 server	h	
	[76, AI'18]	✗	◐	HE	✓	✗	✗	✓	✓	✗	✗	data owners + 1 server	a	uses absolute distance
	[18, ASIACCS'21]	✓	◐	ASS+GC	✓	✓	✓	✓	✓	(✓)[4]	✗	2PC/Outsourcing	a	
HC	[77, SDM'06]	✗	◐	HE+ASS+GC	✓	✓	✗	✓	✗	✓	✗	2PC	h	
	[50, TKDE'07]	✗	◐	blinding *or* SKC	✓	✓	✗	✓	✓	✗	✗	data owners + 1 server	h	SKC not semantically secure
	[49, TDP'10]	✗	◐	HE+GC	✓	✓	✗	✓	✓	✓	✗	2PC	h	
	[48, ISI'14]	✗	N/A	HE	✓	✗	✗	✓	✓	✓	✗	2PC	v	
	[78, ISCC'17]	✗	◐	HE	✓	✓	✗	✓	✗	✗	✓	2PC	v/h	
	[19, ArXiv'19]	✓	◐	HE & GC	✓	✓	✓	✓	✗	✓	✓	2PC	h	
BIRCH	[79, SDM'06]	✗	◐	HE+ASS	✓	✓	✗	✓	✗	✗	✗	2PC	v	
	[80, ADMA'07]	✗	◐	HE+ASS	✓	✓	✗	✓	✗	✗	✗	2PC	a	

1 Of the parameters hold by the respective data owner.
2 Assuming max. 1 party deviates from the protocol.
3 Leaks partial information about cluster sizes.
4 Not implemented, but possible.
5 Can be used with any security model of GCs.

Table 3. History overview of privacy-preserving clustering using secure computation techniques. Privacy indicates if fully privacy protection according to the ideal functionality for privacy-preserving clustering (§3.1) is provided (✗: leakage; ✓: no leakage). ◐ is the semi-honest security model, ● is the malicious security model, N/A indicates that no security model was defined. HE is homomorphic encryption, ASS additive secret sharing, SSS Shamir's secret sharing, GC garbled circuits, OPE oblivious polynomial evaluation, PKC public-key cryptography, SKC symmetric-key cryptography, ZKP zero-knowledge proof, blinding is the use of random values for blinding, and other types of secret sharing are summarized by SS. v indicates that the data that shall be clustered is vertically distributed, i.e., the data owners hold the values for a subset of parameters from all data records. h indicates horizontally partitioned data where the data owners hold complete data records with all parameters, and a is arbitrary data partitioning. L1 leaks intermediate centroids, L2 intermediate cluster sizes, L3 other intermediate values (e.g., intermediate cluster assignments or distance comparison results), and L4 the number of clustering iterations. O1 outputs the final cluster labels/assignments, O2 outputs the final centroids, and O3 outputs the final dendogram/tree structure. The schemes with the best privacy guarantees are marked in bold (we do not consider the number of clustering iterations as a severe leakage as it can be easily avoided, cf. §3.2). The efficient and fully private schemes that we implemented and benchmarked in §4 are highlighted in gray.

46, 48, 57, 71–74, 78, 79, 82, 85, 87–89, 91, 93]. An arbitrary partitioning is a mix of both vertical and horizontal data splitting [8, 12, 22, 76, 80, 83, 90, 100]. A realistic data partition depends on the specific application. We discuss this matter in more detail in §3.3.

Used Secure Computation Techniques. Existing privacy-preserving clustering protocols use two main cryptographic techniques. First, there is a range of works that use homomorphic encryption (HE), e.g., [8, 14, 48, 76, 78, 84, 97, 104], but most of them tend to be relatively slow due to the expensive cryptographic operations. Another research direction uses multi-party computation (MPC) techniques like Yao's Garbled Circuits [21], blinding with random numbers, and secret sharing to achieve better efficiency [11, 16, 18, 44, 45, 72, 83, 94]. These schemes tend to have better runtimes, but higher communication than using HE. However, some MPC techniques [10] also have to rely on non-collusion assumptions between (a subset of) the computing parties which can make them more difficult to deploy in real-world applications. Other protocols use a mix of these techniques aiming at combining the strengths of both approaches [12, 19, 46, 77, 79, 103].

Privacy. As discussed in §1, information leakage can cause severe privacy infringement. Ideally, no information beyond what can be extracted from the final output should be derivable (cf. §3.1). However, most of the proposed privacy-preserving clustering schemes leak intermediate values like the intermediate centroids [43, 84, 85, 88, 94, 97, 98, 100, 105, 108], cluster assignments [43, 57, 58, 71, 83, 86, 87, 89–91, 97, 98, 100, 102–106, 108], and/or cluster sizes [8, 57, 71] in each clustering iteration of K-means or K-medoids, thus, failing to provide full privacy protection. Similarly, both private GMM schemes [44, 45] leak the intermediate covariance matrices, means, and probability values for each Gaussian distribution. Many schemes originating from a round-based clustering algorithm such as K-means or GMM leak the number of clustering iterations until convergence, e.g., [12, 13, 15, 43–45, 82, 83, 94, 100]. However, this issue can be avoided by clustering for a fixed number of iterations independent of the input which must be large enough to reach a good clustering result. However, this results in a longer runtime as more iterations are done than normally with a convergence check. Also most DBSCAN-based and HC-based schemes leak information, e.g., distances between data records [46, 50], the comparison results of distances [48, 72, 75, 79, 80], cluster assignments [22, 46, 47, 49, 73, 75, 76, 79, 80], cluster sizes [22, 47, 73, 75, 76], or may even leak concrete input records for specific data constellations [73, 77] to at least one of the involved parties (independent of the party's data ownership). All in all, we only identified ten clustering protocols shown in Tab. 1 that provide fully privacy guarantees (maximally leaking the number of clustering iterations): [9, 11–19].

Efficiency. As stated before, homomorphic encryption-based protocols such as [14, 48, 76, 78, 84, 97, 104] tend to be computationally expensive and, thus, slower than MPC-based schemes, e.g., [11, 18, 44, 45, 72, 94], which require more communication. Due to space limitations, we will focus here on the ten protocols that provide full privacy (cf. §3.1) and compare them in terms of efficiency. Kim and Chang [15] observe an about 2.85× runtime improvement compared to [13] thanks to a more efficient secure comparison. They as well as Bunn and Ostrovsky [12] use the Paillier encryption scheme without any parallelization making it expensive and slow compared to the other more optimised protocols which use, for example, packing or batching of operations [11]. The K-means protocol by Mohassel et al. [11] was experimentally compared to [14] and [18]. It outperforms the K-means protocol by Jäschke et al. [14] by five orders of magnitude on a dataset with 400 elements thanks to an efficient batching and the usage of GC instead of HE. It is also 19× faster on the same dataset than the private DBSCAN protocol of Bozdemir et al. [18], but DBSCAN often achieves significantly better clustering quality [18]. [18] runs about 13 minutes for clustering 500 elements while Zahur and Evans [17] report more than 550 minutes for their private DBSCAN protocol with 480 records. [18] is also 194× faster than the fully-private affinity propagation protocol by Keller et al. [16] thanks to the use of optimized combinations of GC and ASS. No direct comparison between [9, 11, 19] was done so far.

Choice for Benchmarks. To summarize, the MPC-based protocol of Mohassel et al. [11] is the most advanced private K-means scheme w.r.t. privacy and efficiency. Meng et al. [19] and Cheon et al. [9] provide the only schemes that offer fully private single and complete linkage HC/Mean-shift. Bozdemir et al. [18] propose the most efficient fully privacy-preserving DBSCAN protocol. In §4, we focus on these four works by comparing their computation and communication efficiency, security and privacy, and clustering quality.

3.3 Private Clustering Applications

Privacy-preserving clustering can be generally used for two main purposes: to protect sensitive data when outsourcing the computation and storage and/or when multiple data owners provide input data to the clustering. For each of these scenarios, we discuss a few example applications in the following to give a guideline on how to choose appropriate private clustering protocols.

Example Applications. Multiple data owners who jointly cluster their combined data is an instance of *multi-party computation* (MPC). In the financial market, clustering is used to automatically detect correlations between securities' stock prices in pair trading, i.e., for investment strategies that leverage discrepancies between typically correlated securities [109, 110]. Additionally, it is used for outlier detection to identify credit card, insurance, or tax frauds and insider trading [3, 111]. In this context, it is typically necessary to cluster data from several sources like competing (investment) banks or insurance companies to detect suspicious behavior [112]. Thereby, the different entities might hold information about the same customer, i.e., they have vertically partitioned data. Furthermore, clustering can be used by companies for enhancing marketing measures, e.g., by market segmentation or personalization of recommendation systems [113, 114]. A larger database increases the quality and reliability of the result but business secrets and customer data must be protected. In such scenarios, a horizontal data partitioning where the companies provide data from different customers is more plausible. Additionally, clustering is also used in medical research and diagnosis [115, 116]. In this context, using data from several sources, e.g., several hospitals, reduces potential bias caused by demographics, ethnicities, or cultures.

MPC. For the aforementioned applications, the parties can always safely trust themselves. Thus, full threshold security (i.e., security against up to $N-1$ collusions, cf. §2.2) as provided by some MPC techniques [117, 118] would be an interesting option. Unfortunately, only Keller et al. [16] and schemes based on a threshold secret sharing (like SSS (cf. §2.2)) with the respective threshold can offer this. Additionally, again only Keller et al. [16] provide full privacy guarantees against malicious adversaries (cf. §2.2). However, if the other partners involved are generally trusted but strictly regulated by data protection laws like HIPAA or GDPR, hindering them from directly sharing data, a 2PC- or MPC-based clustering protocol with honest majority and secure against semi-honest adversaries, e.g., [9, 11, 19], might be sufficient and provides significantly better efficiency than MPC techniques that are secure against full threshold semi-honest or malicious adversaries [119].

Outsourcing. While running "generic" MPC protocols is the most straightforward approach to securely cluster on the joint database of data owners, it suffers from high computation and communication costs and might be practically infeasible for a large number of data owners as MPC protocols often scale quadratically in the number of parties. A more efficient alternative can be to outsource the evaluation.[2] Thereby, the data owners (e.g., competing companies conducting market analyses) might prefer to not rely on a non-collusion assumption needed for MPC-based protocols such as [11] where multiple providers must be found and trusted not to collude. Hence, in such a situation an HE-based outsourcing scheme like [9, 14] might be advantageous. Additionally, in an outsourcing scenario, no data owner should be required to be online or actively involved in the computation. This can only be achieved by some protocols: [11, 13–16, 18, 19, 37, 43, 58, 90, 92, 100, 102, 106].

Privacy vs. Efficiency. Furthermore, there exists a trade-off between data leakage and efficiency. Schemes that can leak complete data records (e.g., [73, 74]) should not be used. When generally trusted parties like hospitals are involved, leaking less critical information like the number of iterations (cf. §3.2) might be acceptable to reduce runtime. However, as pointed out in §1, it is not always possible to fully understand and anticipate the effects of leaking intermediate results. Generally, MPC-based protocols are considered to be faster but require more communication than HE-based protocols. We will give more insight on the efficiency of the four most efficient fully private clustering schemes in §4.

Algorithm Characteristics. Finally, another important aspect for choosing the right private clustering scheme are the input parameters. For instance, K-means, K-medoid, GMM, and HC require the number of clusters as input. This is not an issue when the number of clusters is fixed by the application, e.g., if the goal is to split bank customers' behavior into benign and suspicious. However, a more fine-grained analysis might be needed when different types of malicious behavior can occur that significantly differ from each other (e.g., credit card, tax, or insurance fraud), but it is unclear in advance how many untypical behaviors can occur. Clustering might even be used to detect and differentiate these outliers in the first place. In such a case, a protocol that originates from affinitey propogation, DBSCAN or Mean-shift should be chosen as they will flexibly detect the number of clusters. Furthermore, some algo-

2 A single data owner might of course also outsource clustering.

rithms like K-means and GMM can only detect clusters of convex shapes, while DBSCAN can detect arbitrarily shaped clusters [41]. If new data arrives regularly, it might be beneficial to avoid recomputing the complete clustering by using K-means-, K-medoid-, DBSCAN-, or GMM-based protocols (e.g., [8, 11, 43, 44, 46]). Then, different private clustering schemes give different outputs, e.g., centroids [8, 11] or dendograms [19]. For example, a medical analysis detecting typical characteristics for a specific disease should output centroids as they represent these characteristics. Additionally, centroids also allow to assign new data to the created clusters later on which is not possible with only the cluster labels.

To conclude, the following aspects need to be examined when deciding upon a private clustering protocol: scenario (MPC vs. outsourcing), security/privacy requirements, trust level among the data owners, data distribution and splitting, and the plaintext clustering algorithm's characteristics (e.g., required input parameters that can be anticipated in advance). Based on this information, our extensive summary in Tab. 3 can help to choose an appropriate protocol.

4 Evaluation

In this section, we compare the clustering quality (§4.1), security and privacy (§4.2), and efficiency (§4.3) of the four most efficient fully private clustering protocols [9, 11, 18, 19] identified in §3.2. Details about these protocols are provided in Appx. B.

Software Details. We implemented all four protocols in C++17 and instantiate all cryptographic building blocks with a security level of 128 bits. We instantiate all algorithms with optimal parameters to assess their performance assuming perfect conditions. All our implementations are single threaded for a fair comparison of the efficiency of protocols.

MPC-KMeans [11]. In the remainder of this work, we call the private K-means protocol by Mohassel et al. [11] MPC-KMeans. We use the publicly available implementation[3] from the authors of [11] with default parameter values. Specifically, the statistical security parameter is $\lambda = 40$, the computational security parameter $\kappa = 128$, and the bitlength $\ell = 32$.

HE-Meanshift [9]. We call the private Mean-shift protocol by Cheon et al. [9] HE-Meanshift. The implementation uses the HEAAN library [120] with the same parameters as [9] providing 128-bit security. Specifically, the degree of the polynomial modulus of the plaintext ring N_c is set to 2^{17} and the ciphertext modulus q_L is set to 2^{1480}. Thus, the number of plaintext slots in each ciphertext is 2^{16}. Unless explicitly stated, we set the degree parameter for the kernel $\Gamma = 6$, the MinIdx degree parameter $t = 5$, and the Inv iteration parameter $\zeta = 5$.

PCA/OPT [19]. We call the baseline private HC protocol by Meng et al. [19] PCA (complete linkage) and its extension OPT (single linkage). The implementation uses the ABY framework [10] for Yao's garbled circuits and the libpaillier library [121] for Paillier with identical parameters as [19]. Specifically, the symmetric-key security parameter is $\kappa = 128$ bits and the size of the RSA modulus in Paillier encryption is $\kappa_{\text{pub}} = 2048$ bits. The statistical security parameter is $\lambda = 40$.

ppDBSCAN [18]. We call the privacy-preserving DBSCAN protocol by Bozdemir et al. [18] ppDBSCAN. We use the publicly available C++ implementation[4] provided by the authors which is based on the ABY framework [10]. The computational security parameter is set to $\kappa = 128$ bits and the bitlength $\ell = 32$ bits. The parameter $maxIterations$ was set to 4 as also done in [18].

4.1 Clustering Quality

In this section, we evaluate the clustering quality of the four fully private clustering protocols.

Datasets. We use nine datasets from the well-known FCPS [122] and Graves [123] collections designed for benchmarking clustering algorithms. They also include the ground truth separation [124]. In Tab. 4, we summarize four of these datasets for which we present the results of the quality evaluation. The results of our evaluation on the remaining 5 datasets are given in Appx. D.

Metrics for Clustering Quality. We measure the quality of the clustering result using *clustering quality indices.* As no single index is superior [125], we use four well-known indices: Adjusted Rand Index (ARI) [126], Adjusted Mutual Information (AMI) [127], Silhouette Index (SI) [128], and Calinski-Harabasz Index (CHI) [129]. SI and CHI measure the output clusters' separation and compactness while ARI and AMI

3 https://github.com/osu-crypto/secure-kmean-clustering

4 https://encrypto.de/code/ppDBSCAN

Dataset	N	d	K	Property
Hepta	212	3	7	Well-defined clusters
Lsun	400	2	3	Different shapes
Chainlink	1000	3	2	Non-linearly separable clusters
Dense	200	2	2	Different cluster variances

Table 4. Datasets used for evaluating clustering quality, where N is the dataset size, d is the dimension of the data records, and K is the number of clusters.

compare the output clusters to the known ground truth to evaluate the clustering quality [125].

The results for the algorithms with random initialization, MPC-KMeans and HE-Meanshift are averaged over 10 runs. The iterative algorithms MPC-KMeans, HE-Meanshift, KMeans++, and Mean-shift are run for 20 iterations. The number of clusters K for the K-Means and HC protocols, i.e., MPC-KMeans and PCA/OPT, is set to the number of clusters in the ground truth. HE-Meanshift modifies the Mean-shift algorithm to run the mean-shift process on a small and random subset of datapoints, called dusts, to improve efficiency. The number of dusts is set to the largest power of 2 greater than or equal to the number of clusters, to ensure efficiency while maintaining the quality of the clustering.

Original vs. Private Algorithm. We also compare the differences in clustering quality between the private clustering protocols and the original plaintext algorithms to evaluate the *error arising by using privacy preserving techniques*. PCA, OPT and ppDBSCAN are identical to plaintext HC with complete and single linkage and DBSCAN respectively which is why we do not include the results for their plaintext implementations here. The underlying computations in MPC-KMeans are identical to the standard K-means protocol except for differences in the initialization of the centroids. We evaluate this effect using the plaintext KMeans++ [130] algorithm (a variant of K-Means with an improved cluster initialization where the centroids are initialized with data records far apart from each other). HE-Meanshift, in contrast, introduces several modifications to the standard Mean-shift algorithm [53] to make the computation HE friendly, e.g., using a polynomial kernel and adopting dust-sampling for efficient mode-seeking. We compare the clustering quality of HE-Meanshift to a plaintext implementation of Mean-shift to evaluate the combined effect of the changes.

Fig. 1 summarises the results of our evaluation of the clustering quality with the four quality indices.

Hepta Dataset. All algorithms achieve a relatively good clustering quality. PCA, OPT and ppDBSCAN output exactly the ground truth and achieve the best scores on the four indices. MPC-KMeans has a slightly worse clustering quality than the HC algorithms. HE-Meanshift achieves significantly lower scores and its *high standard deviation* in comparison to KMeans++ and Mean-shift indicates that the initialization of dusts has a significant impact on the clustering quality.

Lsun Dataset. OPT and ppDBSCAN output exactly the ground truth and, thus, achieve the maximal scores for the ARI and AMI. While the output of PCA significantly differs from the ground truth, it is noteworthy that it achieves similar scores as OPT and ppDBSCAN on the SI and CHI that measure internal cluster properties, i.e., separation and compactness. HE-Meanshift again shows large standard deviations due to the random initialization. The poor clustering quality achieved by MPC-KMeans and KMeans++ on the ARI and AMI is due to the non-convexly shaped clusters in the ground truth where K-means does not work well. The best scores achieved by HE-Meanshift are significantly higher than their plaintext counterparts, possibly due to favourable (random) initialization for the non-convexly shaped clusters in some of the runs.

Chainlink Dataset. OPT and ppDBSCAN have the same output as the groundtruth and achieve the highest ARI and AMI scores though the presence of non-linearly separable clusters in the dataset leads to lower SI and CHI scores. The remaining algorithms perform poorly. A poor clustering quality of MPC-KMeans and KMeans++ is expected since K-Means does not work well on non-linearly separable clusters.

Dense Dataset. MPC-KMeans, HE-Meanshift, ppDBSCAN, KMeans++, and Mean-shift achieve good scores on all indices while the HC protocols, i.e., PCA and OPT, have significantly lower scores in comparison. Intuitively, the poor clustering quality of HC-based protocols can be attributed to the large variance in one of the clusters of the Dense dataset. This can cause incorrect merging of clusters since clusters are merged based on their proximity, which can be large when the variance of the cluster is high.

Conclusion. ppDBSCAN consistently achieves the highest scores and is able handle different shapes, non-linear clusters, and high cluster variance well. PCA and OPT achieve a relatively good clustering quality on three out of four datasets, but they (completely) fail on the Dense dataset. The K-Means and Mean-shift protocols have comparable clustering quality that heavily varies between different datasets. The K-means-based protocols can only cluster very specific datasets that do not contain non-convexly shaped and non linearly-separable clusters.

HE-Meanshift tends to have large standard deviations which indicate a strong dependency on dust initialization. However, the highest score achieved by HE-Meanshift is comparable to that of plaintext Mean-shift which indicates that the modifications introduced for its HE-friendly computation do not decrease accuracy. In contrast, MPC-KMeans has a small standard deviation and achieves a similar clustering quality to KMeans++, which shows that the randomness used for centroid initialization has a smaller impact on final output.

4.2 Security & Privacy

In this section, we discuss the security and privacy of the four clustering protocols.

Security Model w.r.t Scenario. All four works are in the static semi-honest security model i.e., the adversary can corrupt some of the parties at the onset of the computation and correctly follows the protocol description, but attempts to learn information about the private inputs of the honest parties.

MPC-KMeans, PCA/OPT, and ppDBSCAN consider the outsourced two-party computation setting where multiple data owners secret share their input among two non-colluding servers to privately cluster the dataset. In contrast, in HE-Meanshift, a *single* data owner outsources its computation to a *single* server.

Informally, a protocol is said to be secure if anything that can be computed by a party participating in the protocol can also be derived from the input and output of this party. This is formalized by using a *simulator* which generates a view that is indistinguishable from a real protocol execution given the party's input and output [132]. MPC-KMeans [11] and PCA/OPT [19] provide such a formal proof of security.

The security of HE-Meanshift [9] follows directly from the security of the used CKKS encryption scheme since only the input and final output are sent. We note that the recent attack on the CKKS scheme by Li and Micciancio [133] does not affect the security of HE-Meanshift, as discussed by Cheon et al. [134]. Specifically, the attack requires access to a decryption oracle which is not available to the server in the outsourced single-server computation setting.

Similarly, the security of ppDBSCAN [18] follows directly from the security of the employed secure two-party computation techniques, specifically GC and ASS (cf. §2.2), as no intermediate values are opened and the conversions are provably secure [10].

Leakage from Outputs. Provable security of the protocols ensures that the computation does not leak anything more than what is revealed in an ideal world where a trusted third party obtains the inputs, computes the clustering functionality and returns the output. However, the information leaked from the clustering *output* is not captured in the security definition and we discuss this in the following.

HE-Meanshift outputs the cluster labels for every record in the dataset. However, this is not a privacy concern since the protocol is intended to be used in the outsourced single-server computation setting where the entire dataset is known to the client.

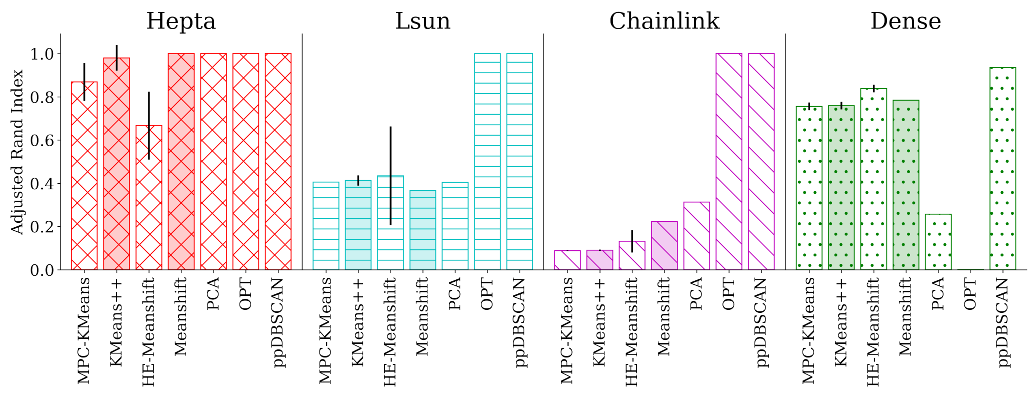
MPC-KMeans and ppDBSCAN can be adapted to output either the cluster centroids or cluster labels. MPC-KMeans also outputs the number of iterations for the clustering to converge which is related to the distribution of the underlying dataset. We have already discussed how to avoid this leakage in §3.2.

The PCA/OPT algorithms output a *point-agnostic dendrogram* in addition to the cluster centroids. The point-agnostic dendrogram is intended to be a privacy-preserving variant of the dendrogram output by a plaintext HC algorithm since the latter provides the complete merging history which leaks information in a setting with multiple data owners. The point-agnostic dendrogram is computed by first applying a random and private permutation on the input records to fuzz the merging history and by retaining the metadata of only sufficiently large clusters. Intuitively, this allows obtaining useful metadata akin to the plaintext computation while still preserving privacy. However, it is unclear how to formalize/measure the information leakage from the protocol output.

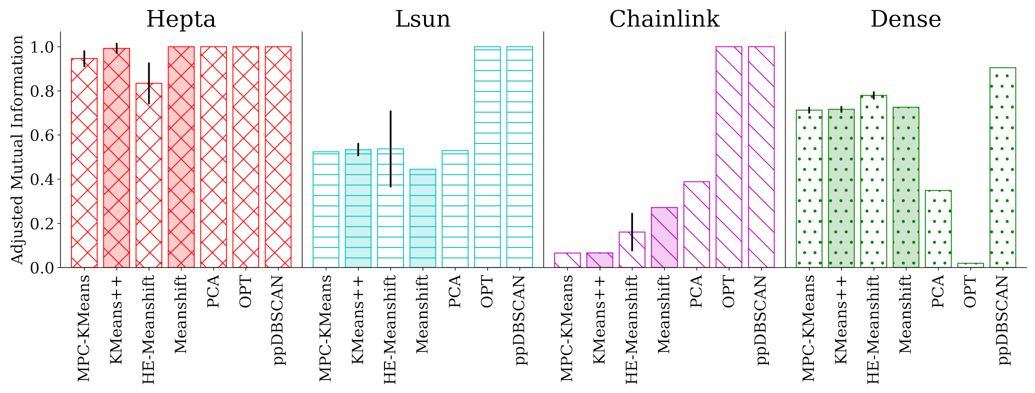
4.3 Efficiency

Asymptotic Analysis. First, we compare the asymptotic runtime, communication, and round complexity of the four investigated private clustering protocols and depict the results in Tab. 5. Asymptotically, MPC-KMeans is the most efficient with respect to communication and runtime in terms of dataset size N, input records' dimension d and number of clusters K.

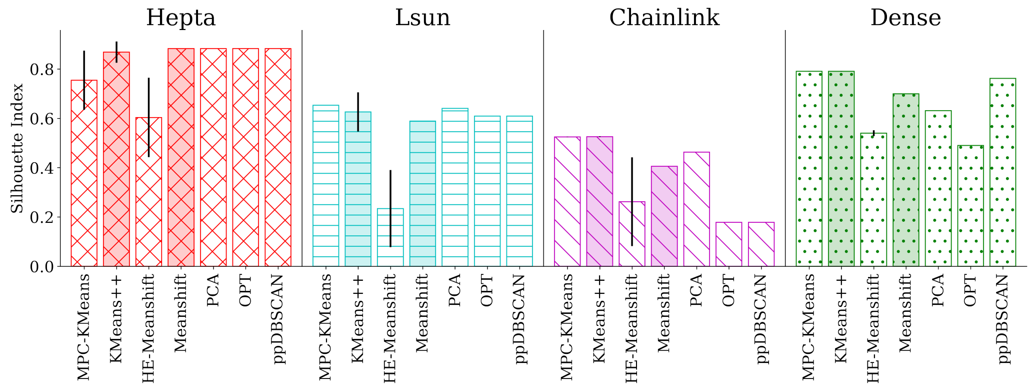
Hardware Details. All experiments are run on two machines (one for each party) each equipped with a 2.8 GHz Intel Core i9-7960X processor running Linux, 32 vCPUs and 128 GB RAM. We consider two network settings. The LAN setting has bandwidth 1 Gbps and



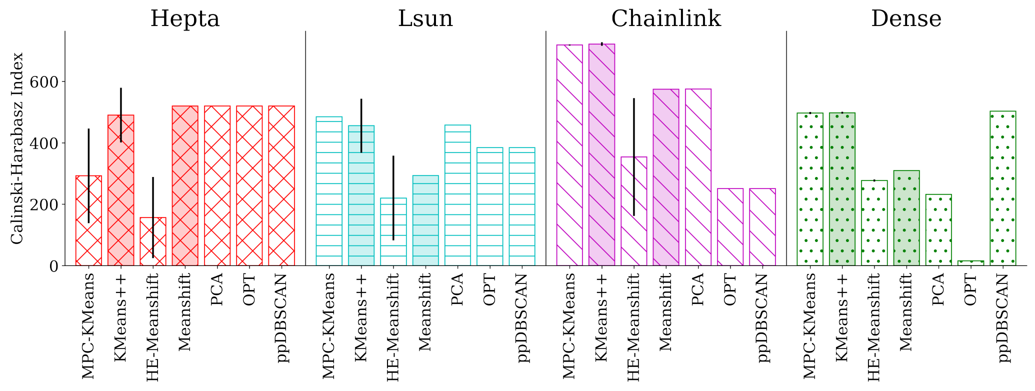
(a) Adjusted Rand Index (ARI)



(b) Adjusted Mutual Information (AMI)



(c) Silhouette Index (SI)



(d) Calinski-Harabasz Index (CHI)

Fig. 1. Clustering quality evaluation of the fully-private clustering protocols MPC-KMeans [11], HE-Meanshift [9], PCA/OPT [19], and ppDBSCAN [18] evaluated on datasets Hepta (red), Lsun (blue), Chainling (purple), and Dense (green). As comparison to the plaintext clustering algorithms (shaded bars), we also include KMeans++ and Mean-shift from Python Scikit-learn [131]. Each subfigure (a-d) corresponds to a different clustering quality index: ARI, AMI, SI, and CHI (cf. §4.1) and larger values indicate better clustering quality. The values are averaged over 10 runs and the error bar shows the standard deviation.

Protocol	Runtime	Communication	Rounds
MPC-KMeans [11]	$\Theta(NK(d+\ell)t)$	$\Theta\left(NK(d\ell^2+\ell\kappa)t\right)$	$\Theta(\lceil\log K\rceil t)$
HE-Meanshift [9]	$\Theta\left((NK_d d^2 t)/(N_c \log d)\right)$	$\Theta(NdK_d\kappa)$	2
PCA [19]	$\Theta(N^3\lambda)$	$\Theta(N^3\lambda\kappa)$	$\Theta(N^2)$
OPT [19]	$\Theta(N^2(\lambda+d))$	$\Theta(N^2(\lambda\kappa+\kappa_{\text{pub}}))$	$\Theta(N^2)$
ppDBSCAN [18]	$\Theta(N^2(N+d))$	$\Theta(N^2\ell\kappa)$	$\mathcal{O}(N^3)$

Table 5. Asymptotic runtime, communication, and round complexity of the private clustering protocols MPC-KMeans [11], HE-Meanshift [9], PCA/OPT [19], and ppDBSCAN [18]. N is the dataset size, d is the dimension, ℓ is the bitlength of the data records, K is the number of clusters, K_d is the number of dusts used in HE-Meanshift, $\kappa = 128$ is the computational security parameter, $\lambda = 40$ is the statistical security parameter, N_c is the number of plaintext slots in CKKS, $\kappa_{\text{pub}} = 2048$ is the size of the RSA modulus in Paillier encryption [65].

RTT 1 ms while the WAN setting has bandwidth 100 Mbps and RTT 100 ms.

Benchmarks. We evaluate the efficiency of the four privacy-preserving clustering protocols and their scalability to large datasets, datasets with many clusters, and multi-dimensional data. We generate synthetic datasets with $N \in \{50, 100, 150, 200, 250\}$ data points of dimension $d \in \{2, 8\}$ and bitlength $\ell = 32$, and number of clusters $K \in \{2, 5\}$. We run the protocols on all possible combinations of the above described parameters. The iterative protocols MPC-KMeans and HE-Meanshift are run for 5 iterations to reach a comparable clustering quality and enabling a fair comparison of their efficiency.

To analyze the scalability of the protocols to large multi-dimensional datasets, we generate a synthetic collection of *large datasets* with parameters $N \in \{2^{13}, 2^{14}, 2^{15}, 2^{16}\}$, $d \in \{1, 2, 4, 8, 16\}$ and $K \in \{2, 5, 10, 15, 20\}$. The memory consumption of MPC-KMeans was too large (greater than 128 GB) to benchmark on datasets where $N \cdot d > 2^{19}$. Similarly, the memory consumption of PCA/OPT [19] and ppDBSCAN [18] was too large even for the smallest dataset with $N = 2^{13}, d = 1$, and $K = 2$ due to the usage of the memory intensive ABY framework [10]. We thus exclude these protocols from our benchmarks on the large datasets. Fig. 4 in Appx. C presents the memory consumption of our implementations of the protocols for a small and large dataset.

Communication. We plot the communication costs in the bottom rows of Fig. 2 (small datasets) and Fig. 5 in Appx. C (large datasets).

The communication cost for HE-Meanshift is identical across different small datasets (Fig. 2) because the entire dataset can be encrypted in one ciphertext. This inefficient packing of the dataset leads to HE-Meanshift's communication being 2× higher than that of MPC-KMeans on average for small datasets. However, for large datasets (Fig. 5), HE-Meanshift's communication cost is up to 11.5× lower than that of MPC-KMeans on average due to optimal packing. The communication cost of PCA is 6× higher than that of ppDBSCAN on average while the communication of ppDBSCAN is 2× higher than that of OPT on average. While the communication cost increases linearly in the input records' dimension d for HE-Meanshift, d does not have a significant effect on the communication of MPC-KMeans since the communication during the assignment of input records to clusters is independent of d. This phase has the highest communication complexity and dominates the overall communication cost. In Fig. 5(f) HE-Meanshift's communication for $K = 10$ and $K = 15$ is identical as both use the same number of dusts $K_d = 16$ and hence send the same number of ciphertexts. The communication costs of PCA/OPT are independent of the input records' dimension d while the communication costs of PCA/OPT and ppDBSCAN are independent of the number of clusters K.

Runtimes. We plot the LAN runtimes in the top row of Fig. 2 (small datasets) and Fig. 5 in Appx. C (large datases), and the WAN runtimes for small datasets in Fig. 3 in Appx. C, all averaged over 10 runs.

MPC-KMeans has the lowest runtime and is up to 700× faster than HE-Meanshift on small datasets and 9.5× faster than HE-Meanshift on large datasets over LAN. Thus, HE-Meanshift scales well to large, multi-dimensional datasets due to lower communication costs, but is not suitable for small datasets. As expected, the runtime of MPC-KMeans in Fig. 5(b) is marginally affected by increasing the dimension of the input records d since its assignment of the input records (which is the bottleneck of the protocol) to the clusters is independent of d. On the other hand, the runtime of HE-Meanshift is linear in d since it directly affects the number of ciphertexts and the efficiency of the bootstrapping operation.

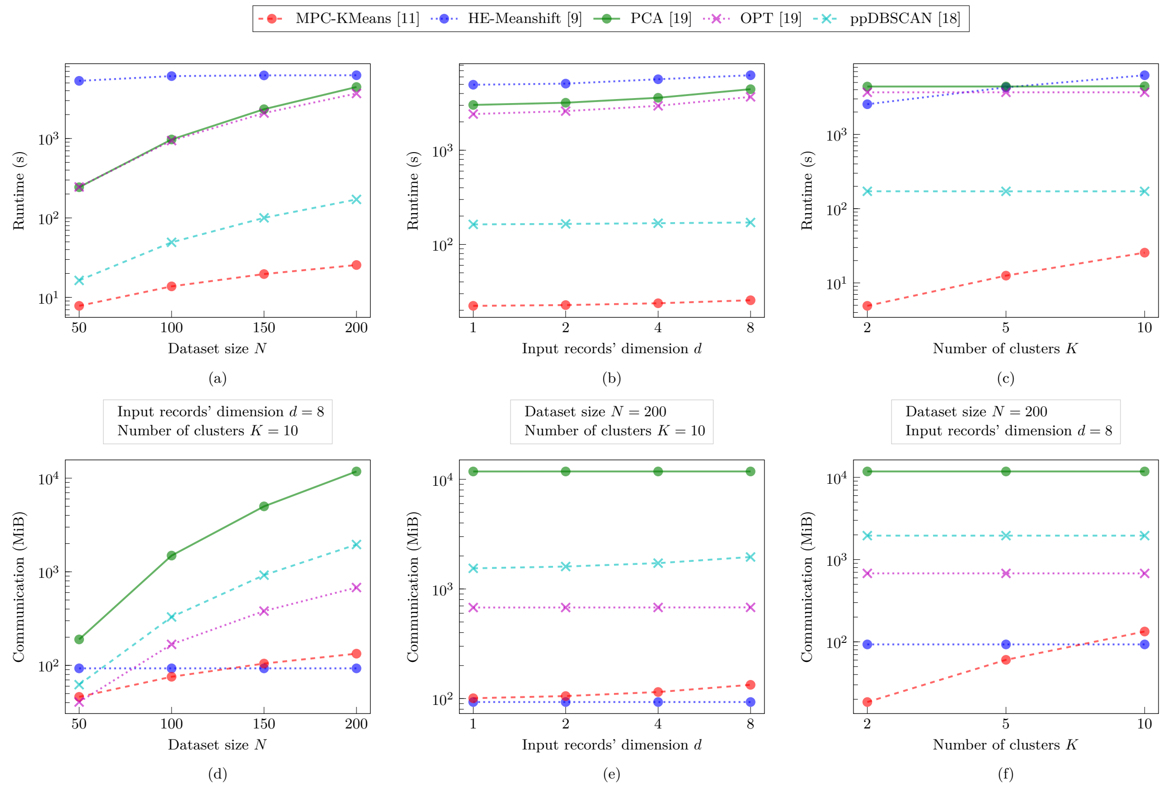


Fig. 2. LAN runtimes in seconds (top row) and **communication** in MiB (bottom row) of the fully-private clustering protocols MPC-KMeans [11], HE-Meanshift [9], PCA/OPT [19], and ppDBSCAN [18] for varying dataset size N, input records' dimension d, number of clusters K, and bitlength $\ell = 32$. In (a) and (d) $d = 8$ and $K = 10$, in (b) and (e) $N = 200$ and $K = 10$, and in (c) and (f) $N = 200$ and $d = 8$.

ppDBSCAN's average runtime is 14× higher over LAN and 2.5× higher over WAN than that of MPC-KMeans. However, since ppDBSCAN's runtime is independent of the number of clusters, this gap in runtime diminishes with the increase in number of clusters. OPT and PCA have similar runtimes wich are 15× higher than that of ppDBSCAN on average over LAN, even though OPT has less communication. Moreover, OPT has the highest runtime on average over WAN. One possible reason for this might be higher concrete computation costs for OPT which is not captured by the asymptotic complexity but highlighted due to benchmarking on small datasets.

5 Real-world Application and Open Challenges

In this section, we discuss the challenges that need to be solved to make privacy-preserving clustering practical for real-world applications.

Parameters. In a privacy-preserving setting, it is typically impossible to perform a preliminary analysis of the data since it is distributed among multiple data owners and not available to a single party. However, to set parameters like the number of clusters K for K-Means (i.e., in MPC-KMeans [11]) and HC (i.e., in PCA/OPT [19]), the distance parameter ϵ for ppDBSCAN [18], or the number of dusts for HE-Meanshift [9], insights about the dataset are often needed to achieve high clustering quality. We also observed in our experimental evaluation that the degree of the kernel and the value of the MinIdx parameter in HE-Meanshift has significant impact on the clustering quality by amplifying the distances between data records. In specific cases, some parameters like the number of clusters K can be given by the application (cf. §3.3). But this is not the case for less intuitive parameters like the initial distribution specifications of GMM or the neighborhood radius in DBSCAN. Only ppDBSCAN [18], out of the four protocols that we benchmarked, can determine these parameters when they are not fixed by the application and inputs are provided by more than one party.

Secure Clustering Quality Evaluation. The issue of dataset-dependent parameters is amplified by the lack of secure clustering quality evaluation techniques. Specifically, plaintext clustering algorithms are often simply run several times with a range of different parameter values and the best output is selected based on the score of a clustering quality index. This is not possible in privacy-preserving clustering. Firstly, as in plaintext clustering, no ground truth is known, so metrics like ARI or AMI that compare to the ground truth cannot be used. Secondly, as the private clustering result is typically split among the data owners or only consists of the centroids, the clusters' compactness and the separation between different clusters cannot be measured without additional secure computation. Thus, also internal indices like SI and CHI cannot be used easily. The inherent overhead of secure computation makes it expensive to perform multiple runs with different parameter values.

Clustering Quality and Efficiency. Clustering algorithms that make minimal assumptions about the shape of clusters and are robust to outliers are especially important in the case of privacy-preserving clustering, because it is impossible to analyze the dataset or remove noisy records before clustering. None of the privacy-preserving clustering protocols we consider investigate soft clustering (cf. §2.1) and only ppDBSCAN has the notion of noise. The K-means-based protocols, i.e., MPC-KMeans, are very sensitive to outliers. Additionally, as shown by our quality evaluation in §4.1 and as discussed in §2.1, K-means only succeeds on clustering convexly shaped clusters which is not the case for all datasets. HE-Meanshift's clustering quality also strongly fluctuates depending on the dataset's properties (cf. §4.1). This is especially problematic for privacy-preserving clustering where the dataset distribution is often not known in advance. In contrast, hierarchical clustering like PCA/OPT is less sensitive to noise and more flexible with respect to the data distribution, i.e., the cluster shapes. However, as shown in §4.3, PCA/OPT cannot be run on large datasets while MPC-KMeans and HE-Meanshift scale significantly better to large datasets. Our evaluation shows that ppDBSCAN performs well on different types of datasets while also having lower runtimes than other protocols (except MPC-KMeans).

Recommendations. Among the protocols we evaluate, MPC-KMeans seems to be the most efficient alternative when clustering large multi-dimensional datasets. HE-Meanshift might be a better choice when a *single* resource-constrained data-owner outsources clustering to a more powerful server over a high-latency and low-bandwidth network. For smaller datasets, ppDBSCAN seems to be the best option which performs well on a variety of dataset types and also achieves low runtimes. However, choosing the input parameter ϵ that determines the maximum distance between two data records to be considered as neighbors requires domain expertise and partial information about the dataset. This can be avoided by using MPC-KMeans which only requires setting the more intuitive number of clusters K which in some cases is also given by the application (cf. §3.3).

Open Challenges. To summarize, for practical application, privacy-preserving clustering protocols must be (1) efficient in terms of runtime and communication, (2) memory efficient, (3) only have parameters that are mostly independent of the input data, (4) insensitive to noise, and (5) flexible to cluster data of any distribution with high quality. Unfortunately, none of the state-of-the-art works can fulfill all these requirements simultaneously. Additionally, there is the need for a secure clustering quality evaluation to assess the quality of a clustering result run in a privacy-preserving manner. Finally, privacy-research has not tackled privacy-preserving soft clustering.

6 Conclusion

In this work, we systematically surveyed and analyzed the state-of-the-art in privacy-preserving clustering. We benchmarked and compared four efficient protocols [9, 11, 18, 19] that securely realize four different clustering algorithms, with respect to clustering quality, communication, and runtime to investigate their practicality for real-world applications. Finally, we discussed open challenges to make privacy-preserving clustering practical.

ACKNOWLEDGEMENTS

We thank Oliver Schick for his support with implementing PCA/OPT [19]. This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). It was co-funded by the Deutsche Forschungsgemeinschaft (DFG) – SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, and by the BMBF and the HMWK within ATHENE.

References

[1] Z. Qian, Z. M. Mao, Y. Xie, and F. Yu, "On Network-level Clusters for Spam Detection." in *NDSS*, 2010.

[2] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and Structural Biotechnology Journal*, 2015.

[3] M. Ahmed, A. N. Mahmood, and M. R. Islam, "A Survey of Anomaly Detection Techniques in Financial Domain," in *Future Generation Computer Systems*, 2016.

[4] F. Masulli and A. Schenone, "A fuzzy clustering based segmentation system as support to diagnosis in medical imaging," *Artificial Intelligence in Medicine*, 1999.

[5] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli, "User profiles for personalized information access," in *The adaptive web*, 2007.

[6] A. Chaturvedi, J. D. Carroll, P. E. Green, and J. A. Rotondo, "A feature-based approach to market segmentation via overlapping k-centroids clustering," *Journal of Marketing Research*, 1997.

[7] C. Gentry and D. Boneh, *A fully homomorphic encryption scheme*. Stanford university Stanford, 2009.

[8] W. Wu, J. Liu, H. Wang, J. Hao, and M. Xian, "Secure and efficient outsourced K-means clustering using fully homomorphic encryption with ciphertext packing technique," in *TDKE*, 2020.

[9] J. H. Cheon, D. Kim, and J. H. Park, "Towards a practical cluster analysis over encrypted data," in *SAC*, 2019.

[10] D. Demmler, T. Schneider, and M. Zohner, "ABY - A framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[11] P. Mohassel, M. Rosulek, and N. Trieu, "Practical privacy-preserving K-means clustering," in *PETS*, 2020.

[12] P. Bunn and R. Ostrovsky, "Secure two-party K-means clustering," in *CCS*, 2007.

[13] F.-Y. Rao, B. K. Samanthula, E. Bertino, X. Yi, and D. Liu, "Privacy-preserving and outsourced multi-user K-means clustering," in *CIC*, 2015.

[14] A. Jäschke and F. Armknecht, "Unsupervised Machine Learning on Encrypted Data," in *SAC*, 2018.

[15] H. Kim and J. Chang, "A privacy-preserving k-means clustering algorithm using secure comparison protocol and density-based center point selection," in *International Conference on Cloud Computing*, 2018.

[16] H. Keller, H. Möllering, T. Schneider, and H. Yalame, "Balancing quality and efficiency in private clustering with affinity propagation," in *SECRYPT*, 2021.

[17] S. Zahur and D. Evans, "Circuit structures for improving efficiency of security and privacy tools," in *IEEE S&P*, 2013.

[18] B. Bozdemir, S. Canard, O. Ermis, H. Möllering, M. Önen, and T. Schneider, "Privacy-preserving density-based clustering," in *ASIACCS*, 2021.

[19] X. Meng, D. Papadopoulos, A. Oprea, and N. Triandopoulos, "Private two-party cluster analysis made formal & scalable," *arXiv:1904.04475v2*, 2019.

[20] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *STOC*, 1987.

[21] A. C.-C. Yao, "How to generate and exchange secrets," in *FOCS*, 1986.

[22] J. Liu, L. Xiong, J. Luo, and J. Z. Huang, "Privacy preserving distributed DBSCAN clustering," in *Transactions on Data Privacy*, 2013.

[23] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow: Secure TensorFlow inference," in *IEEE S&P*, 2020.

[24] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "CrypTFlow2: Practical 2-party secure inference," in *CCS*, 2020.

[25] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *USENIX Security*, 2020.

[26] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2. 0: Improved mixed-protocol secure two-party computation," in *USENIX Security*, 2021.

[27] V. Haralampieva, D. Rueckert, and J. Passerat-Palmbach, "A systematic comparison of encrypted machine learning solutions for image classification," in *PPMLP*, 2020.

[28] F. Boemer, R. Cammarota, D. Demmler, T. Schneider, and H. Yalame, "MP2ML: A mixed-protocol machine learning framework for private inference," in *ARES*, 2020.

[29] L. Song, H. Wu, W. Ruan, and W. Han, "SoK: Training machine learning models over multiple sources with privacy preservation," in *arXiv:2012.03386*, 2020.

[30] H. C. Tanuwidjaja, R. Choi, S. Baek, and K. Kim, "Privacy-preserving deep learning on machine learning as a service—a comprehensive survey," in *IEEE Access*, 2020.

[31] Á. Kiss, M. Naderpour, J. Liu, N. Asokan, and T. Schneider, "SoK: modular and efficient private decision tree evaluation," in *PETS*, 2019.

[32] U. Stemmer, "Locally private K-means clustering," in *ACM-SIAM Symposium on Discrete Algorithms*, 2020.

[33] L. Ni, C. Li, X. Wang, H. Jiang, and J. Yu, "DP-MCDBSCAN: Differential privacy preserving multi-core DBSCAN clustering for network user data," in *IEEE Access*, 2018.

[34] M.-F. Balcan, T. Dick, Y. Liang, W. Mou, and H. Zhang, "Differentially private clustering in high-dimensional Euclidean spaces," in *ICML*, 2017.

[35] D. Su, J. Cao, N. Li, E. Bertino, M. Lyu, and H. Jin, "Differentially private K-means clustering and a hybrid approach to private optimization," in *TOPS*, 2017.

[36] D. Su, J. Cao, N. Li, E. Bertino, and H. Jin, "Differentially private K-means clustering," in *Data and Application Security and Privacy*, 2016.

[37] W. Wu and H. Huang, "A DP-DBSCAN clustering algorithm based on differential privacy preserving," in *Computer Engineering and Science*, 2015.

[38] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016.

[39] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *CCS*, 2015.

[40] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," in *Annals of Data Science*, 2015.

[41] R. Xu and D. Wunsch, "Survey of clustering algorithms," in *TNN*, 2005.

[42] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," in *ACM Computing Surveys*, 1999.

[43] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "PPHOPCM: privacy-preserving high-order possibilistic c-means algorithm for big data clustering with cloud computing," *IEEE Transactions on Big Data*, 2017.

[44] M. Hamidi, M. Sheikhalishahi, and F. Martinelli, "Privacy preserving Expectation Maximization (EM) clustering construction," in *DCAI*, 2019.

[45] X. Lin, C. Clifton, and M. Zhu, "Privacy-preserving clustering with distributed EM mixture modeling," in *Knowledge and Information Systems*, 2005.

[46] I. V. Anikin and R. M. Gazimov, "Privacy preserving DBSCAN clustering algorithm for vertically partitioned data in distributed systems," in *International Siberian Conference on Control and Communications*, 2017.

[47] M. S. Rahman, A. Basu, and S. Kiyomoto, "Towards outsourced privacy-preserving multiparty DBSCAN," in *PRDC*, 2017.

[48] I. De and A. Tripathy, "A secure two party hierarchical clustering approach for vertically partitioned data set with accuracy measure," in *Recent Advances in Intelligent Informatics*, 2014.

[49] G. Jagannathan, K. Pillaipakkamnatt, R. Wright, and D. Umano, "Communication-efficient privacy-preserving clustering," in *Transactions on Data Privacy*, 2010.

[50] A. İnan, S. V. Kaya, Y. Saygın, E. Savaş, A. A. Hintoğlu, and A. Levi, "Privacy preserving clustering on horizontally partitioned data," in *TDKE*, 2007.

[51] H. Steinhaus, "Sur la division des corp materiels en parties," in *Bulletin L'Académie Polonaise des Science*, 1956.

[52] B. S. Everitt, S. Landau, M. Leese, and D. Stahl, "Cluster analysis," in *Wiley*, 2011.

[53] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," in *TIT*, 1975.

[54] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *ICDE*, 1998.

[55] J. M. Pena, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the K-means algorithm," in *Pattern Recognition Letters*, 1999.

[56] Zhexue Huang and M. K. Ng, "A fuzzy k-modes algorithm for clustering categorical data," in *TFS*, 1999.

[57] J. Zhan, "Privacy preserving K-medoids clustering," in *SMC*, 2007.

[58] K.-P. Lin, "Privacy-preserving kernel K-means clustering outsourcing with random transformation," *Knowledge and Information Systems*, 2016.

[59] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, 1988.

[60] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in Large spatial databases with noise." in *SIGKDD*, 1996.

[61] Y. Ren, C. Domeniconi, G. Zhang, and G. Yu, "A weighted adaptive mean shift clustering algorithm."

[62] M. Ester, "Density-based clustering," in *Encyclopedia of Database Systems*. Springer, 2009.

[63] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *ACM SIGMOD*, 1999.

[64] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," in *Journal of the Royal Statistical Society*, 1977.

[65] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999.

[66] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *ASIACRYPT*, 2017.

[67] A. Shamir, "How to share a secret," in *Communication of the ACM*, 1979.

[68] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[69] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *IEEE S&P*, 2017.

[70] S. Kamara and M. Raykova, "Secure outsourced computation in a multi-tenant cloud," in *IBM Workshop on Cryptography and Security in Clouds*, 2011.

[71] S. K. Dash, D. P. Mishra, R. Mishra, and S. Dash, "Privacy preserving K-medoids clustering: An approach towards securing data in mobile cloud architecture," in *Conference on Computational Science, Engineering and Information Technology*, 2012.

[72] A. Amirbekyan and V. Estivill-Castro, "Privacy preserving DBSCAN for vertically partitioned data," in *Intelligence and Security Informatics*, 2006.

[73] K. A. Kumar and C. P. Rangan, "Privacy preserving DBSCAN algorithm for clustering," in *Advanced Data Mining and Applications*, 2007.

[74] W.-j. Xu, L.-s. Huang, Y.-l. Luo, Y.-f. Yao, and W. Jing, "Protocols for privacy-preserving DBSCAN clustering," in *International Journal of Security and Its Applications*, 2007.

[75] D. Jiang, A. Xue, S. Ju, W. Chen, and H. Ma, "Privacy-preserving DBSCAN on horizontally partitioned data," in *International Symposium on IT in Medicine and Education*, 2008.

[76] N. Almutairi, F. Coenen, and K. Dures, "Secure third party data clustering using ϕ data: Multi-user order preserving encryption and super secure chain distance matrices," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 2018.

[77] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A new privacy-preserving distributed K-clustering algorithm," in *SDM*, 2006.

[78] M. Sheikhalishahi and F. Martinelli, "Privacy preserving clustering over horizontal and vertical partitioned data," in *Symposium on Computers and Communications*, 2017.

[79] P. K. Prasad and C. P. Rangan, "Privacy preserving birch algorithm for clustering over vertically partitioned databases," in *Workshop on Secure Data Management*, 2006.

[80] K. Prasad and P. Rangan, "Privacy preserving birch algorithm for clustering over arbitrarily partitioned databases," *ADMA*, 2007.

[81] X. Zhu, M. Liu, and M. Xie, "Privacy-preserving affinity propagation clustering over vertically partitioned data," in *International Conference on Intelligent Networking and Collaborative Systems*, 2012.

[82] J. Vaidya and C. Clifton, “Privacy-preserving K-means clustering over vertically partitioned data,” in *SIGKDD*, 2003.

[83] G. Jagannathan and R. N. Wright, “Privacy-preserving distributed K-means clustering over arbitrarily partitioned data,” in *SIGKDD*, 2005.

[84] S. Jha, L. Kruger, and P. McDaniel, “Privacy preserving clustering,” in *ESORICS*, 2005.

[85] S. Samet, A. Miri, and L. Orozco-Barbosa, “Privacy preserving K-means clustering in multi-party environment,” in *SECRYPT*, 2007.

[86] C. Su, F. Bao, J. Zhou, T. Takagi, and K. Sakurai, “Privacy-preserving two-party K-means clustering via secure approximation,” in *AINA*, 2007.

[87] M. C. Doganay, T. B. Pedersen, Y. Saygin, E. Savaş, and A. Levi, “Distributed privacy preserving K-means clustering with additive secret sharing,” in *International Workshop on Privacy and Anonymity in Information Society*, 2008.

[88] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, “Privacy-preserving user clustering in a social network,” in *Information Forensics and Security*, 2009.

[89] J. Sakuma and S. Kobayashi, “Large-scale k-means clustering with user-centric privacy-preservation,” in *Knowledge and Information Systems*, 2010.

[90] M. Upmanyu, A. M. Namboodiri, K. Srinathan, and C. V. Jawahar, “Efficient privacy preserving K-means clustering,” in *Pacific-Asia Workshop on Intelligence and Security Informatics*, 2010.

[91] T.-K. Yu, D. Lee, S.-M. Chang, and J. Zhan, “Multi-party K-means clustering with privacy consideration,” in *ISPA*, 2010.

[92] M. Beye, Z. Erkin, and R. L. Lagendijk, “Efficient privacy preserving K-means clustering in a three-party setting,” in *Information Forensics and Security*, 2011.

[93] Z. Lin and J. W. Jaromczyk, “Privacy preserving two-party K-means clustering over vertically partitioned dataset,” in *ISI*, 2011.

[94] S. Patel, S. Garasia, and D. Jinwala, “An efficient approach for privacy preserving distributed K-means clustering based on shamir’s secret sharing scheme,” in *Trust Management VI*, 2012.

[95] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, “Privacy-preserving distributed clustering,” in *EURASIP Journal on Information Security*, 2013.

[96] S. Patel, V. Patel, and D. Jinwala, “Privacy preserving distributed K-means clustering in malicious model using zero knowledge proof,” in *Distributed Computing and Internet Technology*, 2013.

[97] D. Liu, E. Bertino, and X. Yi, “Privacy of outsourced K-means clustering,” in *ASIACCS*, 2014.

[98] X. Liu, Z. L. Jiang, S. M. Yiu, X. Wang, C. Tan, Y. Li, Z. Liu, Y. Jin, and J. Fang, “Outsourcing two-party privacy preserving K-means clustering protocol in wireless sensor networks,” in *MSN*, 2015.

[99] S. J. Patel, D. Punjani, and D. C. Jinwala, “An efficient approach for privacy preserving distributed clustering in semi-honest model using elliptic curve cryptography,” *International Journal of Network Security*, 2015.

[100] V. Baby and N. S. Chandra, “Distributed threshold K-means clustering for privacy preserving data mining,” in *ICACCI*, 2016.

[101] Z. Gheid and Y. Challal, “Efficient and privacy-preserving K-means clustering for big data mining,” in *IEEE TrustCom/BigDataSE/ISPA*, 2016.

[102] H. Rong, H. Wang, J. Liu, J. Hao, and M. Xian, “Outsourced k-means clustering over encrypted data under multiple keys in spark framework,” in *Security and Privacy in Communication Networks*, 2017.

[103] K. Xing, C. Hu, J. Yu, X. Cheng, and F. Zhang, “Mutual privacy preserving K-means clustering in social participatory sensing,” in *TII*, 2017.

[104] J. Yuan and Y. Tian, “Practical privacy-preserving MapReduce based K-means clustering over large-ccale dataset,” in *TCM*, 2019.

[105] Z. L. Jiang, N. Guo, Y. Jin, J. Lv, Y. Wu, Z. Liu, J. Fang, S. Yiu, and X. Wang, “Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing,” *Information Sciences*, 2020.

[106] Y. Zou, Z. Zhao, S. Shi, L. Wang, Y. Peng, Y. Ping, and B. Wang, “Highly secure privacy-preserving outsourced k-means clustering under multiple keys in cloud computing,” in *Security and Communication Networks*, 2020.

[107] Y. Wang, “Notes on two fully homomorphic encryption schemes without bootstrapping.” Cryptology ePrint Archive, Report 2015/519.

[108] Y. Cai and C. Tang, “Privacy of outsourced two-party k-means clustering,” *Concurrency and Computation: Practice and Experience*, 2019.

[109] S. M. Sarmento and N. Horta, “Enhancing a pairs trading strategy with the application of machine learning,” in *Expert Systems with Applications*, 2020.

[110] R. Adusumilli, “DBSCAN Clustering for Trading,” 2020, https://towardsdatascience.com/dbscan-clustering-for-trading-4c48e5ebffc8.

[111] S. Panigrahi, A. Kundu, S. Sural, and A. K. Majumdar, “Credit card fraud detection: A fusion approach using dempster–shafer theory and bayesian learning,” in *Information Fusion*, 2009.

[112] A. Sangers, M. van Heesch, T. Attema, T. Veugen, M. Wiggerman, J. Veldsink, O. Bloemen, and D. Worm, “Secure Multiparty PageRank Algorithm for Collaborative Fraud Detection,” in *FC*, 2019.

[113] A. Chaturvedi, J. Carroll, P. Green, and J. A. Rotondo, “A feature-based approach to market segmentation via overlapping k-centroids clustering,” *Journal of Marketing Research*, 1997.

[114] Y. S. Cho, S. C. Moon, S. C. Noh, and K. H. Ryu, “Implementation of personalized recommendation system using k-means clustering of item category based on rfm,” in *ICMIT*, 2012.

[115] Q. Guo, X. Lu, Y. Gao, J. Zhang, B. Yan, D. Su, A. Song, X. Zhao, and G. Wang, “Cluster Analysis: A New Approach for Identification of Underlying Risk Factors for Coronary Artery Disease in Essential Hypertensive Patients,” in *Scientific Reports*, 2017.

[116] F. Masulli and A. Schenone, “A fuzzy clustering based segmentation system as support to diagnosis in medical imaging,” *Artificial Intelligence in Medicine*, 1999.

[117] L. Braun, D. Demmler, T. Schneider, and O. Tkachenko, "MOTION - A framework for mixed-protocol multi-party computation," Cryptology ePrint Archive, Report 2020/1137.
[118] M. Keller, "MP-SPDZ: a versatile framework for multi-party computation," in *CCS*, 2020.
[119] A. Dalskov, D. Escudero, and M. Keller, "Secure evaluation of quantized neural networks," *PETS*, 2020.
[120] "HEAAN," https://github.com/snucrypto/HEAAN, 2020.
[121] "Paillier library," http://acsc.cs.utexas.edu/libpaillier, 2010.
[122] A. Ultsch, "Clustering with SOM," in *Workshop on Self-Organizing Maps*, 2005.
[123] D. Graves and W. Pedrycz, "Kernel-based fuzzy clustering and fuzzy clustering: A comparative experimental study," in *Fuzzy Sets and Systems*, 2010.
[124] M. Gagolewski, "Benchmark suite for clustering algorithms version 1," 2020, https://github.com/gagolews/clustering_benchmarks_v1.
[125] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, 2013.
[126] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, 1985.
[127] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, 2010.
[128] P. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, 1987.
[129] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," in *Communications in Statistics-theory and Methods*, 1974.
[130] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," *ACM-SIAM Symposium on Discrete Algorithms*, 2007.
[131] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *JMLR*, 2011.
[132] Y. Lindell, "How to simulate it–a tutorial on the simulation proof technique," *Tutorials on the Foundations of Cryptography*, 2017.
[133] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers," Cryptology ePrint Archive, Report 2020/1533.
[134] J. H. Cheon, S. Hong, and D. Kim, "Remark on the security of CKKS scheme in practice," Cryptology ePrint Archive, Report 2020/1581.
[135] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, 2007.
[136] X. Liu, M. Yin, J. Luo, and W. Chen, "An improved affinity propagation clustering algorithm for large-scale data sets," in *International Conference on Natural Computation*, 2013.
[137] F. Shang, L. Jiao, J. Shi, F. Wang, and M. Gong, "Fast affinity propagation clustering: A multilevel approach," *Pattern Recognition*, 2012.
[138] D. Dueck, *Affinity propagation: clustering data by passing messages*, 2009.
[139] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, 2014.
[140] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for very large databases," *ACM SIGMOD*, 1996.
[141] C. E. Rasmussen *et al.*, "The infinite Gaussian mixture model," in *NIPS*, 1999.
[142] X. He, D. Cai, Y. Shao, H. Bao, and J. Han, "Laplacian regularized gaussian mixture model for data clustering," *IEEE Transactions on Knowledge and Data Engineering*, 2010.
[143] J. P. Patist, W. Kowalczyk, and E. Marchiori, "Maintaining gaussian mixture models of data streams under block evolution," in *International Conference on Computational Science*, 2006.
[144] R. C. Pinto and P. M. Engel, "A fast incremental gaussian mixture model," *PloS one*, 2015.
[145] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in *EUROCRYPT*, 2018.
[146] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, "Numerical method for comparison on homomorphically encrypted numbers," in *ASIACRYPT*, 2019.

A Discussion of Strengths and Weaknesses of Additional Clustering Algorithms

Affinity Propagation. Affinity Propagation [135] is a deterministic partitioning-based clustering algorithm that has a computational complexity of $\mathcal{O}(N^2t)$ and space complexity of $\mathcal{O}(N^2)$, where N is the dataset size and t the number of clustering iterations [136]. It flexibly determines the required number of clusters based on the input data such that outliers that do not fit into any cluster form a a cluster on their own [135]. The clustering result is independent of the input order of the data records, as Affinity Propagation always iterates through the complete dataset in each iteration. Affinity Propagation requires the input of a *preference* value for each input record that indicates its likelihood to be chosen as exemplar (similar to a centroid in K-means) of a cluster. If the preference values are not well chosen, it can lead to suboptimal clustering results [137]. If all records are equally likely the preference values are set to the same value for all records, e.g., the median or minimum of the distances [135]. The respective distance measure can be freely chosen, thus, also any variable type could be clustered [138]. On the downside, Affinity Propagation can only detect spherical clusters [139] and a re-clustering is needed if new data records are added to the input dataset after it has already been clustered, as this changes the responsibility and availability matrices.

BIRCH. Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is a divisive HC algorithm [140]. Its computation and space complexity is linear in the dataset size [40]. Generally, BIRCH is relatively insensitive to noisy elements as it allows to remove elements in sparse regions [40, 140]. Because each input record is processed incrementally and inserted into the subtree representing the assigned closest cluster, BIRCH can handle new data records well, but is affected by the input order [40, 140]. Moreover, it can only detect convexly shaped clusters with records with metric attributes [40, 140]. Additionally, BIRCH requires to input a threshold for the maximal cluster size and the branching factor of the tree. If the number of cluster K is not given, all sub-clusters in the tree are returned.

GMM. Gaussian Mixture Models (GMM) Clustering is a distribution-based clustering algorithm that uses the Expectation-Maximization (EM) algorithm [64, 141]. GMM has a computational complexity of $\mathcal{O}(NKd^3t)$, where N is the dataset size, K is the number of cluster, d is the data dimension, and t the number of clustering iterations and its space complexity is also linear in N [142–144]. The assumption of a Gaussian distribution of cluster elements restricts the type of the variables to real numbers. GMM fails when clusters have specific constellations, e.g., when one cluster is surrounded by another one, or if the clusters are not convexly shaped. It takes the number of clusters K as input and is relatively sensitive to the selection of the initial parameters of the cluster distributions [41]. Although GMM does not explicitly acknowledge the notion of noise, its result is relatively insensitive to outliers [40, 54], but Keller et al. [16] demonstrate that outliers can still cause significant misclassifcations and incorrect merges between different clusters. As it process the whole data set in each iteration, GMM is not affected by the input order. A few new data records can be clustered by GMM with only a few additional iterations.

B Summaries of Fully-Private Clustering Protocols

MPC-KMeans [11]. Mohassel et al. [11] propose a secure two-party K-means (cf. §2.1) protocol in the semi-honest security model using the ABY framework [10] for secure two-party computation. We call this protocol MPC-KMeans in the following. MPC-KMeans can also be used in an outsourcing scenario [70] where multiple data owners outsource the clustering to two non-colluding servers. The authors also propose a multi-party variant where parties first locally run the plaintext K-means on their local datasets and then proceed to securely compute the joint clustering result based on the previously determined local centroids of all parties.

In the two-party protocol, each data owner runs the plaintext K-means algorithm on its local datasets to compute $\frac{K}{2}$ local clusters. The centroids of these clusters are then secret-shared and used to initialize the centroids for the clustering over the combined dataset.

MPC-KMeans' building blocks are optimized for two computational settings: the amortized setting where the same function is evaluated multiple times on different inputs and the adaptive setting where the inputs to multiple evaluations of the function depend on the output of previous evaluations. Intuitively, the updates of the centroids are non-adaptive in one clustering iteration but adaptive across several iterations. Therefore, the authors introduce efficient protocols for secure multiplication and the calculation of the squared Euclidean distance in the adaptive amortized setting. They also propose an efficient protocol for computing the index of the minimum element in a list of t values using a recursive tree evaluation of a customized Garbled Circuit. This increases the number of rounds by $\lceil \log t \rceil$, but it reduces the communication costs by a factor of 2. MPC-KMeans terminates when the difference between new and old centroids is less than a predefined threshold.

The authors use the squared Euclidean distance. They also benchmark the Manhattan distance ($\underset{i\in[1,d]}{max} |x_i - y_i|$, where d is the dimension) and Chessboard distance ($\sum_{i=1}^{d} |x_i - y_i|$), but show that computing squared Euclidean distance in the adaptive amortized setting is faster than the other two distances. The computation is done on fixed point numbers using the truncation method of [69] where each party locally truncates its share with an error of at most one bit in the least significant bit of the fractional part. The authors show that truncation has a negligible impact on the accuracy of clustering.

HE-Meanshift [9]. Cheon et al. [9] propose a HE-friendly variant of the Mean-shift clustering algorithm (cf. §2.1) in the semi-honest security model using the fully homomorphic encryption (FHE) scheme CKKS [66, 145]. We call this protocol HE-Meanshift in the following. The protocol is designed for the outsourced computation setting where a single, possibly resource-constrained, data owner securely outsources the computation to a server.

CKKS computes on real numbers, but it supports only addition and multiplication. Thus, HE-Meanshift replaces the non-polynomial operations in Mean-shift by polynomial operations. The gradient ascent algorithm used in Mean-shift for mode-seeking requires computing the derivative of the kernel. The authors of [9] propose the HE-friendly polynomial kernel G_{he} shown in Eq. 1. Given the degree parameter $\Gamma \in \mathbb{N}$, the derivative of G_{he} can be computed with a constant multiplicative factor using $\Gamma + 1$ multiplications and 2 subtraction operations.

$$G_{\text{he}}(\boldsymbol{x}, \boldsymbol{y}) = (1 - \|\boldsymbol{x} - \boldsymbol{y}\|^2)^{2^{\Gamma}+1}. \quad (1)$$

HE-Meanshift uses a fixed bandwidth parameter $h = 1$ in the kernel density estimator (KDE) used to compute the density function in Mean-shift. h is a smoothing parameter for the density function and it is the only parameter for the classical Mean-shift algorithm. Although h is fixed in HE-Meanshift, the Γ parameter still offers some flexibility by amplifying the distance between the data points.

Due to the computation overhead of FHE, the authors adopt a random sampling strategy called *dust sampling* which involves sampling K_d points called dusts from the dataset to reduce the $\mathcal{O}(N^2)$ complexity of the original Mean-shift algorithm. HE-Meanshift then performs the mode-seeking on the dusts instead on all points in the dataset. Recall that modes are points corresponding to local maxima in the KDE and represent areas of high density. In Mean-shift, each point is mapped to the cluster containing the closest mode and the number of clusters is equal to the number of distinct modes. Thus, HE-Meanshift can use a relatively low value for K_d that is at least equal to the number of clusters K to compute all clusters in the dataset. This not only improves the efficiency of the mode-seeking but also reduces the costs for bootstrapping, which is now proportional to K_d. However, setting K_d requires prior information about the number of clusters in the dataset in contrast to the plaintext Mean-shift where only a value for h is needed.

After a predefined number of iterations, each point in the input dataset is assigned a cluster label based on the final value of the dusts. Since two or more sampled dusts might converge to the same mode and hence the same cluster, HE-Meanshift uses a secure PointLabeling algorithm to robustly assign records to clusters. The Inv and MaxIdx protocols of [146] are used for division and comparison.

HE-Meanshift does not require communication except from sending and receiving the data to/from the untrusted processing party. It is usable for a single data owner outsourcing the clustering. However, the protocol is not usable for most outsourcing scenarios where multiple data owners cluster their joint data since the encrypted output can be decrypted only by a single data owner.

PCA/OPT [19]. Meng et al. [19] introduce two-party privacy-preserving hierarchical clustering (HC) protocols with single and complete linkage (cf. §2.1) in the semi-honest security model using additively homomorphic encryption [65] and Yao's Garbled Circuits [21]. In contrast to the protocols discussed before in §3, they do not return the resulting clusters/its indices, but a dendrogram (cf. §2.1) indicating the clustering's merging history and metadata containing statistical information about each merge like the new cluster's size and a representative element/centroid. To limit information leakage through this returned metadata, the protocols output only metadata of sufficiently large merges or of the final clusters.

In the baseline protocol, called *PCA*, the two parties calculate the pairwise squared Euclidean distances between the clusters using the additively homomorphic property of Paillier encryption [65]. Then, both parties get access to the plaintext values of the distance matrix blinded with random values such that they can collaboratively cluster the input elements and update the merging dendrogram leveraging two GC-protocols that determine the minimum/maximum distance.

The authors introduce an extension of *PCA* called *OPT* that reduces HC's computation complexity of $O(N^3)$, where N is the dataset size, with single linkage by leveraging the symmetry of the minimum distance. This accelerates the search for the next pair of clusters that have to be merged by a factor of N.

PCA can also be extended to the outsourcing scenario [70] where an arbitrary number of data owners secret share their input data among two non-colluding servers that run the privacy-preserving clustering. However, an extension to more than two servers is not straightforward due to the usage of GCs.

ppDBSCAN [18]. Bozdemir et al. [18] propose a privacy-preserving DBSCAN [60] protocol in the semi-honest security setting using the ABY framework [10]. We call this protocol ppDBSCAN in the following. ppDBSCAN can either be used as secure two-party computation protocol or in an outsourcing scenario with two computing parties, e.g., servers, and an arbitrary number of data owners. The authors point out that the post-processing can be adapted to provide an arbitrary output, e.g., cluster labels, cluster sizes, etc. By assessing the needed recursive depth of the neighborhood exploration ppDBSCAN's complexity can be reduced to a low cubic complexity (from normal cubic complexity). All computations are done on integers.

Initially, the data owners arithmetically share their input records among the two non-colluding parties (which are potentially represented by themselves). Then, the pair-wise squared Euclidean distances are computed between all data records in Arithmetic Sharing [20] to assess which elements have sufficiently many neighbors (i.e., lie in a dense area) to form a cluster. The results are stored as binary values to enhance the the efficiency of the clustering process mostly done with GC [21]. Additionally, the distance computation and the cluster expansion is also parallelized with SIMD operations.

C Additional Benchmarking Results

Fig. 3 summarizes the WAN runtimes of the fully private clustering protocols on small datasets. Fig. 4 depicts the memory consumption of the fully private clustering protocols for a small and large dataset. Fig. 5 summarizes the runtime of HE-Meanshift and MPC-KMeans on large datasets over LAN network.

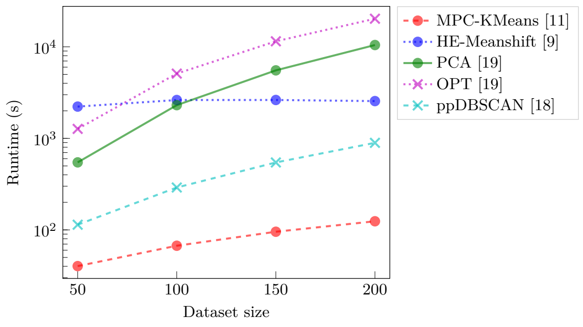


Fig. 3. WAN runtime in seconds of the private clustering protocols MPC-KMeans [11], HE-Meanshift [9], PCA/OPT [19], and ppDBSCAN [18] for varying dataset size N, K=2 clusters, dimension $d = 8$, and bitlength $\ell = 32$.

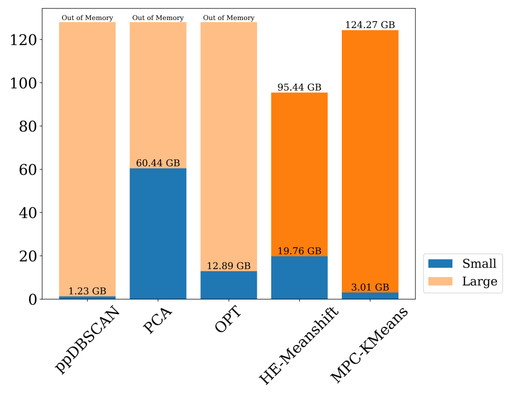


Fig. 4. Memory consumption in GB of the privacy-preserving clustering protocols ppDBSCAN [18], PCA/OPT [19], HE-Meanshift [9], and MPC-KMeans [11] for a small ($N = 200, d = 8, K = 10$) and large ($N = 65536, d = 4, K = 20$) dataset.

D Additional Clustering Quality Evaluation

We compare the clustering quality on nine widely used datasets: Hepta (Tab. 6), Lsun (Tab. 7), Target (Tab. 8), Wingnut (Tab. 9), Tetra (Tab. 10), Chainlink (Tab. 11), and EngyTime (Tab. 12) from [122]; and Dense (Tab. 13) and ZigZag Noisy (Tab. 14) from [123]. Each dataset has different characteristics such as different cluster shapes or different densities.

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.883	519.937
MPC-KMeans	0.869	0.946	0.754	292.539
KMeans++	1.0	1.0	0.883	519.937
HE-Meanshift	0.667	0.834	0.603	156.518
Mean-shift	1.0	1.0	0.883	519.937
ppDBSCAN	1.0	1.0	0.609	384.439
OPT	1.0	1.0	0.883	519.937
PCA	1.0	1.0	0.883	519.937

Table 6. Hepta

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.609	384.439
MPC-KMeans	0.405	0.524	0.653	485.003
KMeans++	0.405	0.524	0.653	485.003
HE-Meanshift	0.434	0.537	0.234	220.118
Mean-shift	0.366	0.445	0.589	293.479
ppDBSCAN	1.0	1.0	0.609	384.439
OPT	1.0	1.0	0.609	384.439
PCA	0.405	0.529	0.641	458.157

Table 7. Lsun

Algorithm	ARI	AMI	SI	CHI
Ground Truth 1	-	-	0.260	27.869
Ground Truth 2	-	-	0.249	0.494
MPC-KMeans	0.534	0.615	0.678	709.651
KMeans++	0.611	0.639	0.742	738.291
HE-Meanshift	0.215	0.311	0.383	101.586
Mean-shift	0.626	0.645	0.766	590.057
ppDBSCAN	1.0	1.0	0.249	0.494
OPT	1.0	1.0	0.249	0.494
PCA	0.207	0.377	0.506	90.502

Table 8. Dataset Target

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.630	1061.016
MPC-KMeans	0.417	0.326	0.567	805.066
KMeans++	0.425	0.334	0.570	815.492
HE-Meanshift	0.475	0.451	0.373	611.832
Mean-shift	0.638	0.538	0.621	1027.873
ppDBSCAN	1.0	1.0	0.630	1061.016
OPT	1.0	1.0	0.630	1061.016
PCA	1.0	1.0	0.630	1061.016

Table 9. Dataset Wingnut

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.726	418.391
MPC-KMeans	0.961	0.975	0.689	390.920
KMeans++	1.0	1.0	0.726	418.391
HE-Meanshift	0.518	0.587	0.124	109.916
Mean-shift	1.0	1.0	0.726	418.391
ppDBSCAN	0.94	0.94	0.694	391.819
OPT	0.000	0.000	-0.436	1.472
PCA	0.987	0.982	0.718	409.221

Table 10. Dataset Tetra

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.179	250.865
MPC-KMeans	0.088	0.065	0.525	718.934
KMeans++	0.087	0.064	0.525	718.788
HE-Meanshift	0.132	0.160	0.262	353.929
Mean-shift	0.223	0.272	0.405	574.488
ppDBSCAN	1.0	1.0	0.179	250.865
OPT	1.0	1.0	0.179	250.865
PCA	0.313	0.388	0.463	575.488

Table 11. Chainlink

Algorithm	ARI	AMI	SI	CHI
Ground Truth 1	-	-	0.557	2921.700
Ground Truth 2	-	-	0.577	3075.082
MPC-KMeans	0.844	0.783	0.578	3158.931
KMeans++	0.843	0.783	0.578	3158.931
HE-Meanshift	0.801	0.720	0.198	1681.223
Mean-shift	0.833	0.769	0.578	3176.809
ppDBSCAN	0.612	0.493	-0.416	115.717
OPT	0.000	0.000	0.479	8.044
PCA	0.042	0.150	0.472	1318.72

Table 12. Dataset EngyTime

Algorithm	ARI	AMI	SI	CHI
Ground Truth	-	-	0.740	433.656
MPC-KMeans	0.756	0.713	0.790	497.182
KMeans++	0.768	0.723	0.790	499.284
HE-Meanshift	0.838	0.779	0.540	277.155
Mean-shift	0.784	0.725	0.699	309.728
ppDBSCAN	0.935	0.904	0.762	503.083
OPT	0.000	0.019	0.490	15.626
PCA	0.257	0.348	0.631	231.817

Table 13. Dense

Algorithm	ARI	AMI	SI	CHI
Ground Truth 1	-	-	-0.050	30.858
Ground Truth 2	-	-	0.500	317.869
MPC-KMeans	0.497	0.636	0.489	321.627
KMeans++	0.519	0.655	0.540	362.227
HE-Meanshift	0.498	0.648	0.538	368.285
Mean-shift	0.542	0.699	0.510	351.971
ppDBSCAN	1.0	1.0	-0.050	30.858
OPT	1.0	1.0	-0.040	30.858
PCA	0.521	0.672	0.504	371.251

Table 14. Dataset ZigZag Noisy

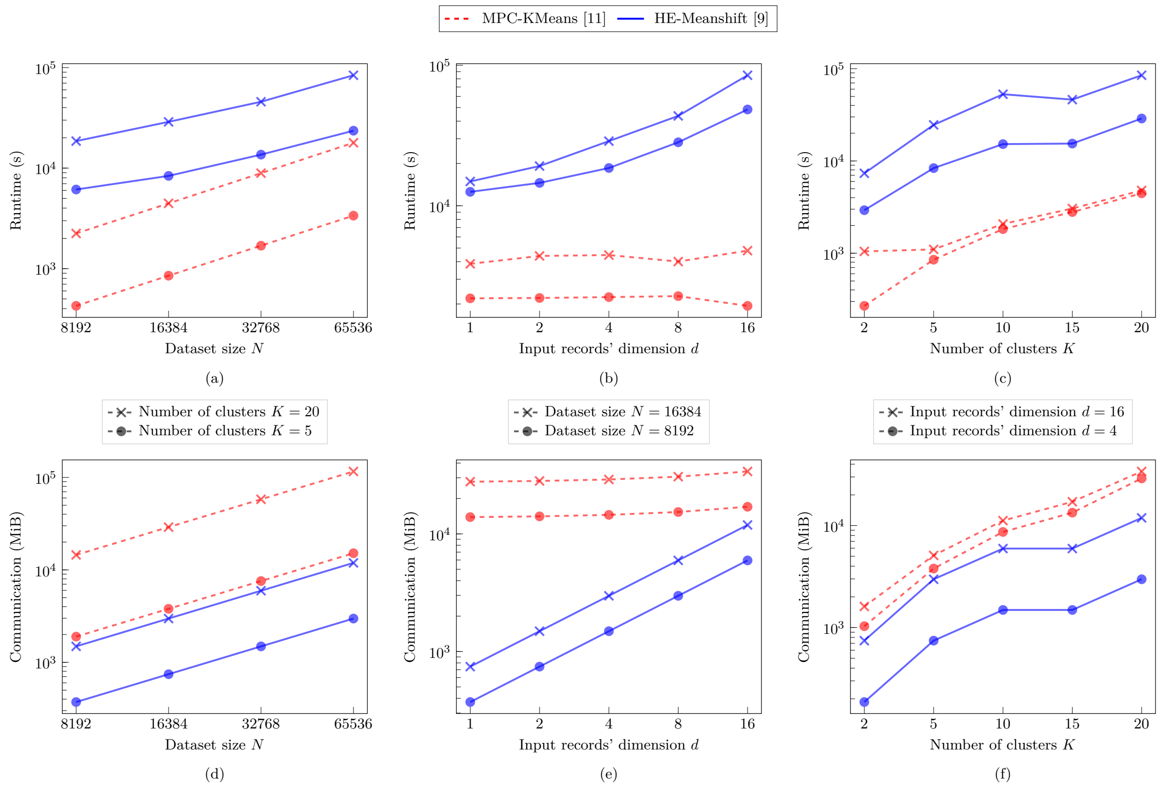


Fig. 5. **LAN runtimes** in seconds (top row) and **communication** in MiB (bottom row) of the fully-private clustering protocols MPC-KMeans [11] and HE-Meanshift [9] for varying dataset size N, input records' dimension d, number of clusters K, and bitlength $\ell = 32$. In (a) and (d) is $d = 4$, in (b) and (e) $K = 20$, and in (c) and (f) $N = 16384$.