CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 0000; **00**:1–26
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Privacy-Preserving Clustering of Unstructured Big Data for Cloud-Based Enterprise Search Solutions

Sm Zobaed and Mohsen Amini Salehi School of Computing & Informatics
University of Louisiana at Lafayette, Louisiana 70504, USA
Email: sm.zobaed1, amini@louisiana.edu

SUMMARY

Cloud-based enterprise search services (*e.g.,* Amazon Kendra) are enchanting to big data owners by providing them with convenient search solutions over their enterprise big datasets. However, individuals and businesses dealing with confidential big data (*e.g.,* criminal reports) are reluctant to fully embrace such cloud services due to valid data privacy concerns. Solutions based on client-side encryption have been developed to mitigate these concerns. Nonetheless, such solutions hinder data processing, especially, data clustering, which is pivotal in applications such as real-time search on large corpora (*e.g.,* big datasets). To cluster encrypted big data, we propose privacy-preserving clustering schemes, called ClusPr, for three forms of unstructured datasets, namely static, semi-dynamic, and dynamic. ClusPr functions based on statistical characteristics of the datasets to: **(A)** determine the suitable number of clusters; **(B)** populate the clusters with topically relevant tokens; and **(C)** adapt the cluster set based on the dynamism of the underlying dataset. Experimental results, obtained from evaluating ClusPr against other schemes in the literature, on three different test datasets demonstrate between 30% to 60% improvement on the cluster coherency. Moreover, we notice that employing ClusPr within a privacy-preserving enterprise search system can reduce the search time by up to 78%, while improving the search accuracy by up to 35%.

KEY WORDS: Cloud trustworthiness; Encrypted clustering; Unstructured big data; Dynamic datasets

1. INTRODUCTION

It is estimated that every day 2.5 Exabytes of both structured and unstructured data are being generated from various sources, such as sensors in weather/traffic/flight systems, organizational documentation/reports, emails, web pages, social media activities (*e.g.,* Facebook and Twitter), digital pictures/videos, and transaction records [23]. The massive volume of generated data is often referred to as *big data*. It is estimated that more than 95% of the big data is in unstructured (*document set*) form [76].

Cloud providers recently offer scalable and convenient *enterprise search* services (*e.g.,* Amazon Kendra [1]) to enable searching over their enterprise big document sets (datasets), stored in the cloud. Such services periodically crawl in the dataset and update their index structure that is used for searching. Specifically, a query is searched against the index and the result-set, referencing the relevant documents with respect to the query, is retrieved for the user. However, using enterprise search services implies outsourcing contents to the cloud that in certain domains, such as law-enforcement and healthcare reports, have raised serious data privacy concerns [38, 21], particularly, after numerous recent data privacy violations [77] in the cloud environments. In one incident [2], more than 14 million Verizon customer accounts information were exposed from their cloud repository in 2018. In another incident [5], confidential information of over three billion Yahoo users were exposed.

An ideal solution for organizations should enable them to securely store their documents in the

cloud while providing real-time enterprise search ability to their authorized users who potentially use thin-clients (*e.g.,* hand-held devices and smartphones) with storage and processing constraints. Client-side encryption [69, 38], in which documents are encrypted with the user's key before outsourcing to the cloud, is a promising method to achieve the desired data privacy. In this method, only the client has the ability to decrypt the documents, hence, it can potentially provide confidentiality against both internal and external attackers [40, 77]. However, the limitation of this method is the lack of processing (*e.g.,* search and clustering) ability on the encrypted documents.

Searchable Encryption systems (*e.g.,* [55, 28, 72, 73, 59]) have been developed to enable privacy preserving search ability over encrypted data. Such systems predominantly extract keywords (aka tokens) from documents to build an encrypted index, which is traversed against a search query at the search time to find relevant documents. The problem arises for big datasets where the index structure is prohibitively large, such that traversing it becomes the search bottleneck [38]. One approach to resolve the bottleneck is topic-based clustering of the index structure, thereby, pruning the search space and limiting it only to the search query context.

However, clustering encrypted data is challenging because the data semantic is lost, once it is encrypted. The clustering challenge gets further complicated when we consider dynamism that exists in some big datasets. We define a *static dataset* as the one that remains unchanged, once it is uploaded to the cloud. Obsoleted data (*e.g.,* HUB5 [13]), archived data (*e.g.,* ACCC, RFC [38]), newsgroup data, book contents are the examples of such datasets. Alternatively, a *dynamic dataset* refers to the ones whose document set grows or shrinks over time, such as criminal records[10], digital libraries [14], and social network feeds [9].

Prior research works (*e.g.,* [38, 71, 22]) suggest that statistical characteristics of keywords that are present in the document set (*e.g.,* keyword co-occurrences across different documents) can be utilized to cluster the keywords of an encrypted document set. Accordingly, in this research, our *goal* is to develop a topic-based clustering mechanism, called ClusPr, for various types of unstructured big datasets with sensitive contents. In particular, considering the dynamism that potentially exists in certain datasets, we propose two variants of ClusPr that are tailored for static and dynamic datasets. Unlike straightforward K-means clustering algorithm [24] where number of clusters and cluster-centers are initialized arbitrarily, we propose two different types of heuristics to cluster static and dynamic datasets. We evaluate the clustering schemes on three different datasets, in terms of the number and coherency of resulting clusters. We deploy our clustering methods within a secure cloud-based enterprise search system, called S3BD [38], and show the advantage of our clustering schemes in improving relevancy and timeliness of the search results.

In sum, the contributions of this research are as follows:

- We propose a method to estimate the suitable number of clusters (K) needed to partition a given encrypted dataset (see §4.1).
- To enable privacy-preserving clustering in the cloud, we develop three clustering schemes (namely, S-ClusPr, SD-ClusPr, and FD-ClusPr) that can cluster encrypted data across static, semi-dynamic, and fully-dynamic unstructured datasets, respectively (see §5.2, §5.3).
- We evaluate and analyze the developed clustering schemes against both encrypted and plain-text clustering approaches with respect to the cluster goodness metrics (see §7.3).

The rest of the paper is organized as follows. In Section 2, we discuss about background study and related prior works. We explain the overview of our proposed system architecture in Section 3. Then, in Section 6, we provide a review on the considered threat model and provide security analysis. We discuss about result comparison and performance analysis in Section 7. Finally, Section 8 concludes the paper.

2. BACKGROUND AND RELATED WORK

2.1. Fundamental Data Clustering Algorithms

Once the number of clusters is determined, a dataset is ready to be clustered by utilizing various clustering algorithms [24, 45, 36, 27, 54]. The tokens are distributed into the cluster with respect to the most similar cluster center. Popular clustering algorithms, such as *K*-means, *X*-means build clusters based on the convergence of center shifting. For a dataset with n datapoints, K clusters, i iterations, and f features, provided that $k \times i \times f \leq n$, the time complexity of *K*-means is $\mathcal{O}(n \times k \times i \times f) \approx \mathcal{O}(n^2)$ [30, 15]. In addition, Elbow method [6], Silhouette coefficient [58], and Bayesian Information Criterion (BIC) scores [67] are commonly utilized to identify the suitable cluster set. These procedures iteratively build different sets of clusters and nominate a final set based on the minimum loss with respect to the clusters' centers. Adopting any of these procedures increase the time complexity by K times. Hence, the total complexity stands to $\mathcal{O}(Kn^2)$ that is not tractable for big data [39, 30]. To avoid this, we propose two sub-tasks: first, from the index structure, we find the potential centers with $\mathcal{O}(n \times c)$ time complexity, where c denotes the maximum number of centers and we have $c << n$; Second, we build a cluster-wise token distribution function to assign each token to a proper cluster. Let q denote the number of appropriate centers (and we have $q \leq c$). Then, the overall time complexity for the two sub-tasks is $\mathcal{O}(2n \times c)$.

The idea of topic-based clustering has been studied and applied extensively on plain-text datasets. Correspondingly, Xu and Croft [71] proposed to build clusters on a homogeneous index (*i.e.,* all of the terms share a nearly similar topic) that improved the effectiveness of a search system compared to standard distributed information retrieval systems. The authors used *K*-means clustering algorithm, and to assign components among the clusters, they used KL-divergence [25]. Then, utilizing the maximum likelihood estimation theory, their proposed method determines the highest relevant cluster based on the incoming search query. However, the overall process is computationally intensive that impacts the real-time search over big data.

Mary and Kumar [47] addressed the challenges of clustering dynamic data (*e.g.,* Twitter and streaming data). They utilized Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [64]. However, DBSCAN falls short in clustering datasets containing different densities, which is common in big data [16]. Alternatively, ClusPr clusters datasets irrespective to their density.

2.2. Privacy-Preserving Data Clustering Schemes

A large body of research has been undertaken to enable processing of the encrypted data (ciphertext). Zhou *et al.* proposed a linear transformation-based solution for matching queries against encrypted data while ensuring data privacy on the cloud without any intervention of the data owner [74]. However, linear transformation methods support secure *K*-nearest neighbor (KNN)-based query matching approaches but not the clustering. This is because clustering is not invariant to linearly transformed data. The optimal linear transformation has a prerequisite of knowing the true cluster means, which is not possible to obtain before generating the cluster [62]. In addition, we assume that the data are tokenized and encrypted before transferring to the cloud. Therefore, unlike [74], where the entirety of encrypted data is queried using time-consuming cryptographic calculations, we use the statistical properties of the data without revealing any meaningful part of it to the cloud. Sun *et al.* proposed a searchable encryption method by forming a tree index structure that operates based on the cosine similarity and $TF \times IDF$ [61, 60] measures. However, the solution is not scalable for big data, because the search index can become large to the extent that it impacts timeliness of the search operation. We believe that our proposed clustering approach can be a complement to [61, 60] where the central index is partitioned topically into multiple small size index structures that can improve the search time and efficiency.

Homomorphic encryption has become a popular method to perform computation over the encrypted data. Several variations of the homomorphic encryption such as fully or partially Homomorphic encryption [34, 70] have been proposed to enable privacy-preserving data processing on the cloud. Zhu *et al.* [75] proposed a secure aggregation and division protocol based on

homomorphic encryption to securely compute clusters without tampering with the privacy of individual peers in a peer-to-peer system. However, their clustering technique does not consider data dynamism. Pang and Wang proposed a homomorphic scheme that provides security to outsourced data uploaded from multiple parties in a twin-cloud system [52] that is assumed to be a semi-honest environment, whereas, we assume cloud to be untrusted in terms of storing/processing sensitive data [44]. Wang *et al.* proposed *HK*-Means++ that combines *K*-Means clustering with finding the suitable cluster numbers [66]. In addition, the work leverages homomorphic encryption scheme to solve the encrypted data manipulation, distance, and convergence calculation. Although our work is comparable to *HK*-Means++, it can only cluster static datasets. Moreover, the experiments were performed only on one dataset and it is not clear how the method performs on other datasets.

We note that the current implementations of the homomorphic encryption technique imply a high computational overhead [32] which affects the real-time response of a search system, particularly, for big datasets [75].

Vaidya and Clifton [65] proposed a solution to cluster encrypted datasets in which different data attributes are stored in distinct storage systems. Then, the clustering was carried out in each one of the data storage systems individually. However, this solution is time consuming and cannot serve the real-time constraint we consider in this work.

2.3. Estimating Number of Clusters

A prerequisite to clustering is to estimate the suitable number of clusters K. However, identifying K is an NP-hard problem [53], hence, a large body of research has been undertaken to provide heuristic methods in which K is approximated [26, 76, 33, 27]. Established clustering methods, such as Silhouette [58], Gap Statistic [63], Stability Selection [43], Consensus [3], and Progeny [37], generate a series of various clustering sets upon considering various number of clusters. Later, they select the optimal clustering set from the series. The common drawback of these methods is the computational overhead, because the clustering has to be performed multiple times with respect to the different number of clusters.

Pelleg and Moore [54] proposed a regularization framework for approximating K upon utilizing X-means clustering. The method is a modified version of K-means clustering that improves cluster distributions by frequently attempting subdivision until it satisfies a predefined stopping criterion. However, lack of prior knowledge about the dataset makes it difficult to apply the associated stopping criterion [36]. In X-means method, an optimization function is used to choose the smallest number of clusters with the maximum possible amount of variation within the dataset. The algorithm starts with K=1 and increment it until the variation reaches a plateau. This starting point is considered as the optimal K for the K-means method [27]. Although the clustering methods are dataset independent, estimation of an appropriate K is generally dataset-specific. K should be higher in sparse data and lesser in comparatively dense data. Likewise, we propose to estimate appropriate K with respect to the content of the considered dataset.

Research Works	Estimating #Clusters	Encryption Approach	Cloud's Trustworthiness	Using Edge Computing	Real-time Support	Dynamic Data Clustering	Multiple Data Owners
Wang *et al.* [66]	No	Homomorphic	Semi-honest	No	No	No	No
Valdiya & Clifton [65]	No	Homomorphic	Semi-honest	No	Yes	No	Yes
Pang & Wang [52]	No	Homomorphic	Semi-honest	No	Yes	No	Yes
Sun *et al.* [61, 60]	No	User-side	Honest-but-curious	No	Yes	No	No
Zhu *et al.* [74]	No	Homomorphic	Honest	No	No	No	Yes
Woodworth *et al.* [38]	No	User-side	Honest-but-curious	No	Yes	No	Yes
ClusPr (proposed)	Yes	User-side	Honest-but-curious	Yes	Yes	Yes	Yes

Table I. Summary of the existing privacy-preserving clustering approaches and positioning our proposed work (ClusPr) with respect to them.

2.4. Positioning of the Proposed Work

Our proposed work is motivated from Woodworth *et al.* method for topic-based clustering on encrypted keywords over the central index using K-means method [38]. The cluster-wise token

distribution function was determined based on the statistical data of each encrypted keyword or token. The authors use a predefined K value. Such K value is inefficient, because the appropriate number of clusters could be varied based on the dataset characteristics. Moreover, as the authors only considered static/unchanged data, the proposed scheme is not capable of processing dynamic data. On the other hand, ClusPr provides a heuristic to approximate the suitable number of clusters and then, clustering the data while maintaining the data privacy on the cloud. For a dynamic dataset, where documents are added or removed over time, because of the re-clustering operation, clusters are shrunk or expanded to reflect the dynamism of the dataset.

Table I summarizes the notable related studies in the literature and positions the contribution of this paper with respect to them.

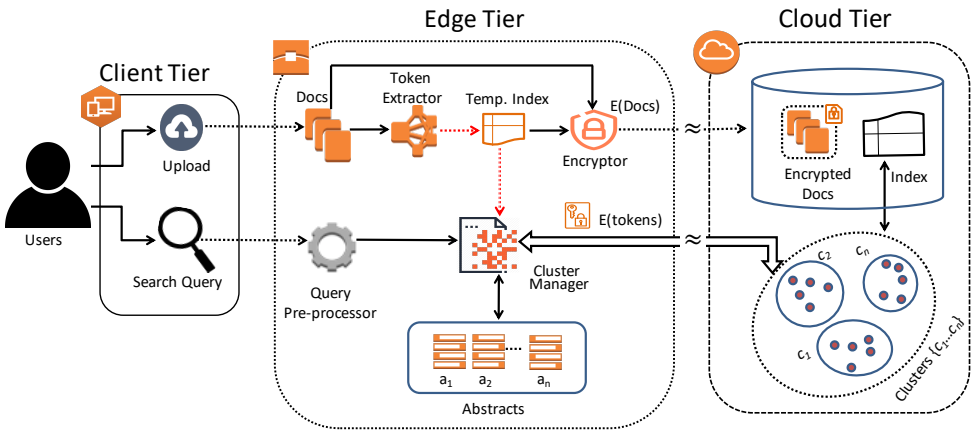


Figure 1. Overview of the context where ClusPr is deployed in a three-tier architecture (of client, edge, and cloud) to facilitate a secure cloud-based search service. The edge tier is assumed to be on the user premises and trusted. It is used to ease the computational overheads imposed by privacy and clustering related processes.

3. THREE-TIER ARCHITECTURE OF CLUSPR

Figure 1 presents an architectural overview of the context where ClusPr is developed. The architecture represents applying ClusPr for S3BD, a cloud-based secure semantic search system that requires clustering over encrypted data [38]. The architecture represents a three-tier system based on a client device, edge system, and the central cloud system. The edge tier resides on the user's premises (hence, is considered trusted) to relieve the client tier from processing computationally intensive tasks. This is particularly important for non-static (*i.e.,* semi-, fully-dynamic) datasets where documents have to be processed as they are uploaded to the cloud tier over time.

In the specific context of S3BD, upon uploading a document by the user, the document is passed through *Token Extractor* on the edge tiers to retrieve the keywords (aka tokens) semantically representing the document. For dynamic datasets, a temporary index structure is used to store the extracted tokens representing the occurrences of each new token in different documents. Next, the document is encrypted by the user's key and is securely stored on the cloud repository. Next, a *Temporary Index* structure is formed based on the extracted tokens of the documents in question before encrypting and uploading them to the cloud. The Temporary Index structure shows the tokens, their frequency, and their appearances across the uploaded batch. Tokens of the Temporary Index are encrypted by the *Encryptor* using the user's key. By encrypting documents as well as the extracted tokens, Encryptor preserves the data privacy on the cloud. Note that, although we can technically use homomorphic encryption to maintain the statistical properties (frequency and co-appearances), for efficiency reasons, in the current implementation, we keep the properties unencrypted. We assume that such properties do not reveal meaningful information about the data.

In fact, in [66], *K*-means clustering was used over homomorphically encrypted big data and showed that the time overhead of clustering can be prohibitively expensive. In the next step, the Temporary Index is fed to the *Cluster Manager* to make the suitable clustering decision on the cloud. Cluster Manager may decide to keep the existing clusters and only update them by the entries of the Temporary Index. Alternatively, upon observing a major update in the Temporary Index, the Cluster Manager decides to exhaustively re-cluster all of the tokens. Though a few of the aforementioned prior works can cluster encrypted data, they fall-short in clustering dynamic datasets, whereas, ClusPr can cluster both static and dynamic data while ensuring privacy. We explain the updating and re-clustering procedures ClusPr in Section 5. Cluster Manager is also in charge of generating and maintaining *Abstracts*. Each abstract a_i is a sampled summary of a corresponding cluster C_i on the cloud tier [20]. Abstracts are to prune the search operation and navigate the search only to clusters that are topically-related to the query. Further details about Abstracts are described in Section 4.4.

For static datasets, the architecture is streamlined such that the extracted tokens are encrypted and directly fed into the Index structure on the cloud tier. Once the data uploading procedure is completed, the cloud tier initiates the clustering procedure. As there is no re-clustering procedures defined for static clusters, the Cluster Manager is only in charge of generating and maintaining the abstracts [20]. It is noteworthy that, in the architecture of Figure 1, the dashed arrows located in the edge tier are to highlight the differences for dynamic datasets.

In the implementation, we chose to use RSA [31], which is a widely-adopted and highly secure deterministic encryption technique, to encrypt the documents and extract their tokens before uploading them to the cloud. Other techniques, such as AES with repeating initialization vector (*e.g.,* AES-SIV), can be used for encryption as well. It is noteworthy that probabilistic encryption techniques [41], such as AES-CBC/GSM [17], cannot be used, because they generate different ciphers for the same token.

The index has a key-value structure, where each key is a token and its corresponding value is a set of pairs, each one representing the document that includes the token and the frequency of that particular token. Note that both the keys and document names are encrypted. For this reason, in the current implementation, we chose to maintain frequency values in the plain-text format. Further details of the proposed static and dynamic data clustering schemes are presented in Section 4 and 5 respectively.

Cluster C_i on the cloud tier includes a group of index entries that are topically similar. As the indexed tokens are encrypted and do not carry any semantic, topic-based clustering of them is a challenging task. We hypothesize the tokens that are semantically similar tend to appear in the same documents. Accordingly, clustering is performed based on the relative frequency of tokens' co-occurrences across all the documents. The union of *K* topic-based clusters ($C_1 ... C_k$) is equivalent to the index. Upon issuing a search query by the user, the abstracts with the highest similarity to the search query are identified. Then, only the clusters associated with the abstracts are searched.

4. PRIVACY-PRESERVING CLUSTERING SCHEME FOR STATIC BIG DATASETS (S-CLUSPR)

In this part, first (in Section 4.1), we elaborate on how to estimate the appropriate number of clusters that should be formed to represent a static big dataset. Second, in Section 4.2, we provide an algorithm to form the center of each cluster. Then, in Section 4.3, we explain methods to distribute the indexed terms across clusters. Finally, in Section 4.4, we describe the way pruning is achieved, *i.e.,* the method that navigates a search query to relevant cluster(s).

4.1. Estimating the Number of Clusters for Static Big Datasets

Depending on the characteristics of a dataset and distribution of tokens in its documents, the appropriate number of clusters (K) can vary significantly. However, optimally determining K directly impacts the accuracy of topic-based clustering and, subsequently, the efficiency of the

system (*e.g.,* search application) that uses the clusters. Encrypted tokens and their metadata, including documents they appear in and their frequency, are the only available parameters to estimate K. The tokens and their metadata are generated by a keyword extractor that retrieves n single or multi-phrase tokens from each document. We assume that all documents are treated equally and the value of n is the same across all documents in a given static dataset.

Estimating K for the static dataset is performed based on the following two steps: (1) building Token-Document Frequency Matrix; and (2) constructing Normalized Matrix.

Step-1: Building Token-Document Frequency Matrix. To be able to follow the scheme, we consider an example using *five* tokens and *six* documents in Table II. We initialize a token-document matrix $\mathbf{A}$ from the index structure. In the matrix, each row represents a token and each column represents a document. Although our approach does not deal with plain-text tokens, just for further readability, in the Table II, we redundantly show the plain-text tokens (in "Word" column) along with their encrypted forms (in "Hash" column). Each entry $a_{i,j}$ of matrix $\mathbf{A}$ represents the frequency of i^{th} token in j^{th} document (denoted as $f(i,j)$).

Table II. Token-Document Frequency Matrix $\mathbf{A}$, built based on the index structure

Word	**Hash**	$\mathbf{d_1}$	$\mathbf{d_2}$	$\mathbf{d_3}$	$\mathbf{d_4}$	$\mathbf{d_5}$	$\mathbf{d_6}$
Book	**Uh5W**	30	0	23	4	40	0
Solve	**/Vdn**	5	0	0	60	34	0
Traffic	**oR1r**	0	23	0	30	0	0
Net	**vJHZ**	52	49	0	23	0	26
Enter	**tH7c**	0	45	68	0	3	5

For a big dataset, the matrix size can be prohibitively large and sparse. To avoid this, we trim the matrix to include only the tokens that are influential in building clusters. We define *document co-occurrences* as the number of documents containing a particular token. Then, to build the token-document frequency matrix $\mathbf{A}$, we only take into account tokens whose document co-occurrences are either greater than or equal to the mean value of the document co-occurrences across the whole dataset.

Step-2: Constructing Normalized Matrix. To make the relationship among tokens and documents quantifiable and comparable, we need to normalize the token-document frequency matrix. Considering that $a_{i,j}$ represents the strength of association between token t_i and document d_j, the maximum value in column j of the token-document frequency matrix represents the token with the highest association with document d_j. Hence, for normalization, we divide the value of each entry of $\mathbf{A}$ to the highest value in the corresponding column of the matrix and the result is stored in a new matrix, called matrix $\mathbf{N}$. The value for each entry $n_{i,j}$ is formally calculated based on Equation 1.

$$n_{i,j} = \frac{a_{i,j}}{\max\limits_{\forall i} a_{i,j}} \tag{1}$$

Step-3: Building Probabilistic Matrices $\mathbf{R}$ and $\mathbf{S}$ The goal, in this step, is to calculate the topic similarity among encrypted tokens. For that purpose, we need to calculate the probability that topic of a token shares similarity with other tokens. We hypothesize that tokens that co-occur across documents are likely to share the same topic. Besides, the magnitude of similarity between two tokens could be influenced by the tokens' distribution across the dataset. For instance, specific terms appear only in a few documents and are not widely distributed throughout the dataset. Such sparsely distributed tokens have low co-occurrences with other tokens which increases the diversity of topics in a dataset and potentially raises the required number of clusters (K). We leverage the normalized matrix ($\mathbf{N}$) to perform a two-phase probability calculation that yields a matrix (denoted as $\mathbf{Q}$) representing token-to-token topic similarity. In the first phase, we calculate the *importance* of

each token to each document. The importance of token t_i, in document d_j, denoted as $\tau_{i,j}$, is defined based on Equation 2.

$$\tau_{i,j} = \frac{n_{i,j}}{\sum_{\forall k} n_{i,k}} \tag{2}$$

Considering Equation 2 and matrix $\mathbf{N}$, we generate matrix $\mathbf{R}$ whose entries represent the importance of each token across all documents. In fact, each entry $r_{i,j}$ of $\mathbf{R}$ represents the probability of choosing a document d_j, having token t_i. That is, $r_{i,j} = \mathbb{P}(t_i, d_j)$.

In the second phase, we calculate the importance of each document to each token. The importance of document d_j for term t_i, denoted by $\delta_{j,i}$ and is defined based on Equation 3.

$$\delta_{j,i} = \frac{n_{j,i}}{\sum_{\forall q} n_{q,i}} \tag{3}$$

Considering each $\delta_{j,i}$ and $\mathbf{N}$, we generate $\mathbf{S}$ whose entries represent the importance of each document with respect to each token. In fact, each entry $s_{i,j}$ represents the probability of choosing t_i from d_j (*i.e.,* we have $s_{i,j} = \mathbb{P}(d_j, t_i)$).

Step 4- Constructing Matrix Q to Determine the Number of Clusters Recall that $\mathbf{R}$ is a token-to-document matrix and $\mathbf{S}$ is a document-to-token matrix. To identify the similarity among the encrypted tokens, we multiply $\mathbf{R}$ and $\mathbf{S}$. As the number of columns and rows of $\mathbf{R}$ and $\mathbf{S}$ are equal, it is possible to multiply matrix $\mathbf{R}$ with $\mathbf{S}$. The resultant matrix, denoted as $\mathbf{Q}$, is a token-to-token matrix and serves as the base to determine the number of required clusters. Each entry $q_{i,j}$ denotes the topic similarity between token i and j. More specifically, $q_{i,j}$ indicates the magnitude to which token i shares similar topic with token j for $i \neq j$ and is calculated as $q_{i,j} = \sum_{\forall i,j} r_{i,j} \cdot s_{j,i}$. Table III shows matrix $\mathbf{Q}$ for the example we discuss in this section.

Table III. Cluster decision matrix $\mathbf{Q}$ is built based on the multiplication of $\mathbf{R}$ and $\mathbf{S}$ matrices

Word-Hash	Book Uh5W	Solve /Vdn	Traffic oRir	Net vJHZ	Enter tH7c
Book- Uh5W	0.39	0.25	0.01	0.18	0.09
Solve- /Vdn	0.26	0.45	0.12	0.12	0.02
Traffic- oRir	0.02	0.26	0.21	0.33	0.18
Net- vJHZ	0.10	0.07	0.08	0.58	0.15
Enter- tH7c	0.09	0.01	0.08	0.28	0.37

Diagonal entries of $\mathbf{Q}$ signify the topic similarity of each token with itself and dissimilarity (*i.e.,* separation) from other topics. More specifically, the value of $q_{i,i}$ indicates the magnitude that term t_i does not share its topic with other terms. Therefore, we define diagonal entries ($q_{i,i}$) as *separation factor*, because for each token, it represents the token's tendency to stay separate from other topics. As such, summation of the separation factors can approximate the number of clusters (K) needed to partition topics of a dataset. Let m denote the total number of tokens in $\mathbf{Q}$. Then, Equation 4 is used to approximate K for a given dataset. We use the ceiling function to make K an integer value.

$$k = \lceil \sum_{i=1}^{m} q_{i,i} \rceil \tag{4}$$

Correctness of K is verified using a hypothesis that states K for a set should be higher if individual elements of the set are dissimilar, otherwise K should be low [26, 29]. Equation 4 is the core of approximating K. According to this equation, the maximum K value can reach to M, when the

documents are highly distinct and each individual token of the documents represents a unique topic, otherwise it is lower than M. Hence, our approach conforms with the clustering hypothesis.

4.2. Center Selection

In *K*-means clustering, generally, the clusters' centers are arbitrarily chosen [19, 45]. Then, based on a distance measure function (*e.g.,* Euclidean distance [19] or semantic graph [45]), dataset elements are distributed into the clusters. *K*-means operates based on iteratively shifting clusters' centers until it converges. However, we realized that the extremely large number of tokens make the iterative center shifting step (and therefore *K*-means clustering) prohibitively time-consuming for big data [18]. Accordingly, in this part, we are to propose a big-data-friendly method to cluster encrypted tokens.

The key to our clustering method is to dismiss the iterative center shifting step. This change entails initial clusters' centers not to be chosen arbitrarily, instead, they have to be chosen proactively so that they cover various topics of the dataset. For that purpose, a naïve method can choose the top *K* tokens that have the highest number of associated documents. Although this approach chooses important (highly associated) tokens, it ends up selecting centers that have a high topical overlap. We propose to choose tokens that not only have high document association but also cover diverse topics exist in the dataset.

We define *centrality* of a token i, denoted Φ_i, as a measure to represent a topic and relatedness to other tokens of the same topic. Assume that tokens are sorted in a descending manner, based on the degree of document association. Let U represent the union of documents associated to the currently chosen centers. Also, for token i, let A_i represent the set of documents associated to i. Then, *uniqueness* [38] of token i, denoted ω_i, is defined as the ratio of the number of documents associated to i but not present in U (*i.e.,* $|A_i - U|$) to the number of documents associated to i and are present in U (*i.e.,* $|A_i \cap U|$). Uniqueness indicates the potential of a token to represent a topic that has not been identified by other tokens already chosen as centers. Particularly, tokens with uniqueness value greater than 1 have high association to documents that are not covered by the currently chosen centers, hence, can be chosen as new centers.

Recall that each entry $q_{i,j}$ of matrix $\mathbf{Q}$ represents the topic similarity between tokens i and j. Besides, diagonal entry $q_{i,i}$ measures separation of token i from others. Therefore, the total similarity token i shares with others can be obtained by $\Sigma_{\forall j|j\neq i} q_{i,j}$. Note that for token i, we have $\Sigma_{\forall j} q_{i,j} = 1$, hence, the total similarity for token i is equal to $1 - q_{i,i}$. Centrality of a token is measured by the uniqueness of the token, the magnitude of similarity the token shares with others, and the magnitude of it being isolated. That is, for token i, centrality is defined as: $\Phi_i = \omega_i \times q_{i,i} \times (1 - q_{i,i})$.

Algorithm 1 shows the high-level pseudo-code to select maximum of K centers from the set of indexed tokens of a dataset. In addition to K, the algorithm receives the central index and the $\mathbf{Q}$ as inputs. The algorithm returns a set of at most K center tokens, denoted $centers$, as output. In the beginning, the output set is initialized to null. U represents the set of documents covered with the chosen centers. A heap structure, denoted Θ, is used to store a pair for each token and its centrality value. For each token i, the uniqueness and centrality values are calculated (Steps $5 - 13$) and the corresponding pair is inserted to the heap. Note that tokens with uniqueness lower than one do not have the potential to serve as a cluster center. In the next step, we select at most K center tokens that have the highest centrality values.

4.3. Distributing Encrypted Tokens Across Clusters

Once K tokens are nominated as cluster centers, the remaining tokens of the index are distributed across the clusters with respect to their *relatedness* (a.k.a. distance) with the center tokens.

Because there is no intersection between the non-center tokens and members of the $centers$ set, we can model the token distribution across the clusters as a weighted bipartite graph where the weight of each edge represents the relatedness between a token and a center. Figure 2 depicts an example of a bipartite graph to show the relationship of each token and centers. Solid lines show the edge with the highest weight for each token that represent the cluster that a token should be

ALGORITHM 1: Pseudo-code to determine clusters' centers

Input : K, $\mathbf{C}$ matrix, and $Index$ (with tokens sorted descendingly based on the degree of document association)
Output: $centers$ set that includes at most K center tokens

```
Function Choose Center(k, Q, Index):
    centers ← ∅
    U ← ∅
    Θ ← {(∅, ∅)} //Pairs of tokens and centrality values
    foreach token i ∈ Index do
        ω_i ← CalculateUniqueness(i, U)
        if ω_i > 1 then
            A_i ← CalculateDocumentAssoc(i, Index)
            U ← U ∪ A_i
            Φ_i ← (ω_i × q_{i,i} × (1 − q_{i,i}))
            Add pair (i, Φ_i) to max-heap Θ based on Φ_i
        end
    end
    centers ← Extract K max pairs from Θ heap
    return centers
end
```

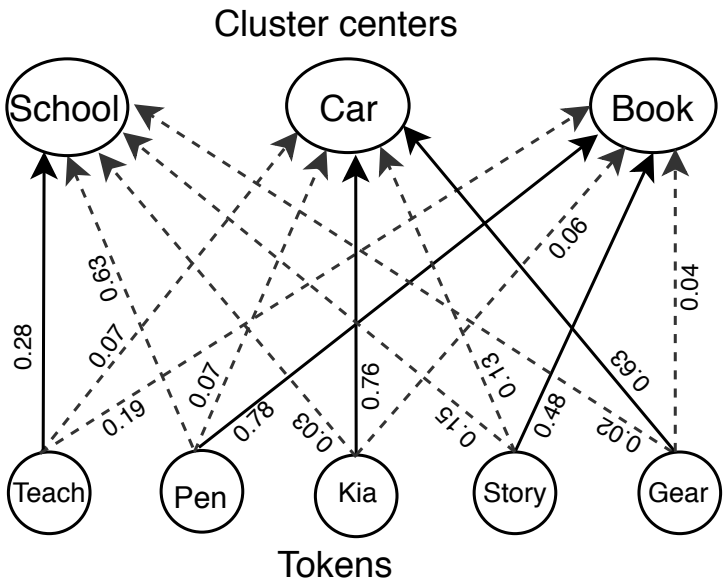


Figure 2. A bipartite graph representing the relatedness among centers and remaining tokens. The weight of each edge represents the relatedness of a token and a center. Solid lines show centers that offer the maximum relatedness for a token.

distributed to. Established techniques (*e.g.,* semantic graph [45], Euclidean distance [19]) are to calculate the relatedness, however, these methods are not appropriate for encrypted tokens that are sparsely distributed [19] [45].

As encrypted tokens lose their semantics, we ought to define the relatedness between tokens based on their statistical characteristics and then leverage it to distribute each token to the cluster that offers the maximum relatedness.

Intuitively, the relatedness measure between tokens t_i and t_j, denoted $r(t_i, t_j)$, is defined based on the magnitude of their *co-occurrences*, *i.e.,* the number of documents where the two tokens appear together [38, 76]. Let F_i and F_j respectively denote the sets of documents that t_i and t_j are appeared in. Then, the intuitive co-occurrence of the two tokens is $F_{co} = F_i \cap F_j$. However, a deeper analysis

reveals that quantifying the relatedness only based on the cardinality of co-occurrence (*i.e.,* $|F_{co}|$) can be misleading for the two following reasons:

First, intuitive co-occurrence ignores the magnitude of disparity across F_i and F_j that negatively impacts the relatedness between t_i and t_j. The disparity is determined based on the symmetric difference (*i.e.,* we have $F_{dis} = F_i \oplus F_j$). Accordingly, to consider the impact of both co-occurrence and disparity, we define a new measure, called *relative co-occurrence*, and leverage it to determine the relatedness between t_i and t_j.

Second, intuitive co-occurrence ignores the importance of t_i and t_j in each document $d \in F_{co}$. Accordingly, to measure the co-occurrence value in each document d, denoted $\upsilon(t_i, t_j, d)$, we consider the importance of each one of the tokens relative to their importance across all documents of F_{co}. We use frequency of a token in a document to measure its importance in that document. Formally, in document d, we calculate the value of co-occurrence based on Equation 5.

$$\upsilon(t_i, t_j, d) = \frac{f(t_i, d)}{\sum\limits_{\forall m \in F_{co}} f(t_i, m)} \cdot \frac{f(t_j, d)}{\sum\limits_{\forall m \in F_{co}} f(t_j, m)} \tag{5}$$

Similarly, we utilize Equation 6 to measure the impact of disparity between two tokens in each document $d \in F_{dis}$, denoted $\varphi(t_i, t_j, d)$.

$$\varphi(t_i, t_j, d) = \frac{f(t_i, d)}{\sum\limits_{\forall m \in F_{dis}} f(t_i, m)} + \frac{f(t_j, d)}{\sum\limits_{\forall m \in F_{dis}} f(t_j, m)} \tag{6}$$

In document d, once we know the co-occurrence and disparity between t_i and t_j, we can calculate the relative co-occurrence as $\rho(t_i, t_j, d) = \upsilon(t_i, t_j, d) - varphi(t_i, t_j, d)$. Then, the relative co-occurrence across all documents of the two tokens (*i.e.,* $F_i \cup F_j$) is leveraged to calculate the relatedness between them.

Assuming c as the token that represents center of a given cluster (*i.e.,* $t_i = c \in centers$), we define relatedness between c and token t, according to Equation 7. Token t is distributed to the cluster whose center offers the maximum relatedness. Note that, in this equation, to emphasize the importance of token t in document d, we also consider its frequency ratio.

$$r(c, t) = \sum_{d \in (F_t \cup F_c)} \rho(t, c, d) \cdot \frac{f(t, d)}{\sum\limits_{\forall m \in F_t} f(t, m)} \tag{7}$$

4.4. Pruning Clusters to Expedite the Search Operation

The purpose of building topic-based clusters is to achieve scalable search over big data via limiting (pruning) the search scope based on the query topic, instead of exhaustively traversing the whole index structure. For pruning, we need to identify the clusters that are semantically relevant to the search query and discard the irrelevant ones. However, pruning is a challenging task when we operate on the encrypted data in the cloud.

To overcome the challenge, we require the topic of each cluster in plain-text, such that we can identify the clusters whose topics are semantically related to the search query and only consider those clusters for searching. For that purpose, in our previous work [20], we established a method to represent the topic of each cluster C_x (denoted α_x) by considering the top-n most-frequent tokens of C_x. The tokens of α_x are decrypted and maintained on the edge tier of ClusPr in a structure called *Abstract*. Abstracts are leveraged to measure the topic similarity between a query and their corresponding clusters. In the next step, the search is conducted on the clusters that are most relevant to the query. For further details about creating abstracts and pruning operation, interested readers can refer to our earlier study [20, 38].

5. PRIVACY-PRESERVING CLUSTERING SCHEME FOR DYNAMIC BIG DATASETS (DYNAMIC CLUSPR)

5.1. Overview

In the previous section, we explained clustering of static (*e.g.,* archive) encrypted big datasets. However, many big datasets are dynamic (*e.g.,* healthcare data, criminal records) [47] and their contents change over time. In this section, we deal with clustering and subsequently searching over such datasets. We consider two types of dynamic datasets: First is the *semi-dynamic* datasets whose contents are updated in batch over time (*e.g.,* Museum of Modern Art (MoMA) dataset [14]); Second is *fully-dynamic* datasets whose contents are constantly updated (*e.g.,* Twitter streams [12]).

The latest changes on the dataset have to be reflected in the clusters. Otherwise, altered documents are not retrieved by the search system, even if they include relevant contents. In fact, the updates on the dataset affect the tokens' co-occurrences and, subsequently, the clustering arrangement. As such, the challenge is to know *how the addition or deleting documents change the topics and number of clusters*.

Given the size of big datasets, reconstructing clusters (called *re-clustering*) upon arrival of every single document or a small batch of documents is time-prohibitive. Moreover, the small updates generally cause negligible changes in the co-occurrences of tokens that are unlikely to modify the arrangement of clusters. Only significant updates can cause decisive changes on the magnitude of co-occurrence and relatedness that entail re-clustering. Accordingly, the two followup questions are: *when to perform re-clustering?* and *how to re-cluster the tokens?* To address these questions, based on the type of dynamic datasets, we propose two clustering schemes in ClusPr: Semi-dynamic data clustering scheme (SD-ClusPr) and Fully-dynamic data clustering scheme (FD-ClusPr).

5.2. Semi-Dynamic Data Clustering Scheme (SD-ClusPr)

In semi-dynamic datasets, topic-based clustering can be initially achieved on the first batch of documents in the dataset according to the method described in the previous section. Then, the re-clustering decisions are made depending on the changes caused by the new batch of documents. That is, we need to determine whether the change caused by the extracted tokens of the new batch is significant or not.

To determine the significance of changes caused by the tokens of the new batch, we utilize χ^2 (chi-square) distribution test [7] that can identify significant changes observed in a variable of a given population. The χ^2 test is known as *testing goodness of fit* and it is represented by Equation 8, where O_i is the observed and E_i is the expected value of a particular variable in K trials.

$$\chi^2 = \sum_{i=1}^{k} [(O_i - E_i)^2 / E_i] \tag{8}$$

We consider the number of the extracted tokens in the new batch and the number of tokens in the existing clusters. Our null hypothesis (H_0) is to perform re-clustering and χ^2 test is employed to check the validity of H_0. If the difference between the number of new tokens and existing tokens is small, a low value of χ^2 is obtained. For one degree of freedom with 95% confidence interval, the value of $\chi^2 = 3.841$ fails to reject H_0. Alternatively, if the number of tokens in the new batch is significantly smaller than the number of existing tokens, χ^2 value becomes higher that denotes significant deviation from H_0. Then, the decision is to reject H_0 and keep the existing clusters.

Once the re-clustering decision is made, we use the method explained in Section 4 to cluster tokens of the updated dataset. In the event that re-clustering is not achieved, the new tokens are accumulated with the of tokens of the next batches. As a result, the total number of new tokens becomes significant that leads to a lower χ^2 value and subsequently acceptance of H_0.

Updating Clusters Let U_1 a new batch of documents that introduces a set of new tokens $T = \{t_1, t_2, ..., t_n\}$ that does not exist in the existing clusters. Assume that based on the re-clustering

decision method, mentioned in the previous part, we determine to keep the existing clusters $\{C_1, C_2, ..., C_n\}$ to accommodate T.

To distribute $t_i \in T$ to a cluster, we can measure the relatedness as explained in Section 4.3. Alternatively, we can leverage the set of abstracts $\{A_1, A_2, ..., A_n\}$. As they are in the plain-text format, a more accurate relatedness measurement can be conducted using the semantic similarity, as opposed to inferring the relatedness based on token co-occurrences in documents. In this case, we use Word2Vec [49] model to calculate the relatedness of t_i and abstract A_j. Then, t_i is assigned to a cluster that offers the highest relatedness. To avoid poor assignments, we define θ as the relatedness threshold that should be reached to assign t_i to C_j. In the event that t_i cannot join any cluster, a new cluster, called $C_{new} \in C$, is formed and t_i is considered as its center. The above procedure is repeated for all $t_i \in T$.

ALGORITHM 2: Pseudo-code to update clusters in SD-ClusPr.

Input : set of abstracts A, $tempIndex$, θ
Output: H, map of new tokens to clusters

Function `SD-ClusPr(`$A, tempIndex, \theta$`)` **:**
  $T \leftarrow tempIndex \setminus CentralIndex$
  $H \leftarrow \emptyset$
  $A \leftarrow \cup_{i=1}^{n} A_i$
  $\Phi \leftarrow \emptyset$
  //Max-heap to find the abstract with highest similarity
  foreach *token* $t \in T$ **do**
    foreach $a_{ij} \in A$ **do**
      $s \leftarrow$ sim (a_{ij}, t)
      if $s > \theta$ **then**
        Add (s, i) to Φ
      end
    end
    if $\Phi \neq \emptyset$ **then**
      //Allocate t to existing cluster
      $(t, i) \leftarrow$ Extract max pair from Φ
      Add (t, i) to H
    end
    else
      //Forming a new abstract and cluster and add it to H
      $A_{n+1} \leftarrow \{t\}$
      $A \leftarrow \cup_{i=1}^{n+1} A_i$
      Add $(t, n+1)$ to H
    end
  end
  Encrypt H and push it to the cloud tier
end

Determining the value of θ Threshold We estimate the value of θ threshold by leveraging the abstracts $\{A_1, A_2, ...A_n\}$. Recall that the elements of abstract A_i are the ones that best represent the topic of its corresponding cluster C_i. We define coherency of A_i as the average similarity distance across pairs of its elements. Let $\{a_{i1}, ..., a_{ip}\}$ be the set of elements of A_i. Then, coherency of A_i, denoted K_i, is defined based on Equation 9 where $sim(x, y)$ shows the similarity distance between $(x, y) \in A_i \times A_i$.

$$K_i = \frac{\sum_{\forall (x,y) \in A_i \times A_i | x \neq y} Sim(x, y)}{\binom{p}{2}} \tag{9}$$

Then, we define θ as the global minimum across all abstracts (*i.e.,* $\theta = \min_{\forall i} K_i$). This implies that a new token can join a cluster only if its distance does not worsen the coherency of current clusters. Otherwise, the new token forms its own cluster.

Algorithm 2 shows the pseudo-code of how to update clusters in SD-ClusPr, in case we choose not to perform re-clustering. In addition to the set of abstracts (A) and θ, the algorithm receives the set of tokens for a new document batch, which is stored in form of a temporary index. The algorithm returns the H structure that includes the mapping of new tokens to their respective clusters. In Steps $7-9$, for each new token, we calculate the similarity distance with respect to all abstract elements a_{ij} and check whether the similarity distance exceeds θ or not. If it exceeds θ, we make a pair of similarity distance and corresponding abstract number, denoted as (a_{ij}, t) and build max-heap Φ based on the distance (in Step $10-12$). If Φ contains any value, we extract from it the pair that has the largest value (*i.e.,* the abstract that offers the most topic similarity for t). Then, in Step 17, the pair of (t, i) is added to H. On the contrary, if Φ is null, it implies that no cluster offers a considerable similarity to t, and so, in Steps $19-24$, we build a new abstract and cluster using t. Finally, we encrypt the tokens of H and push it to the cloud tier. On the cloud end, *cluster manager* updates its clusters based on H.

5.3. Fully-Dynamic Data Clustering Scheme (FD-ClusPr)

Unlike SD-ClusPr, for fully-dynamic datasets, clusters have to be formed or updated upon arrival of the documents. That is, continuous or burst arrival of new documents should trigger FD-ClusPr. Accordingly, in FD-ClusPr, we consider two cases in forming clusters: (A) *initial case* that occurs when first document arrives and there is no existing cluster and (B) *update case*, where the existing clusters have to be updated based on the new changes in the dataset.

In the initial case, the edge tier extracts the set of new tokens from the uploaded document(s). We designate the token with the highest frequency to represent the topic and choose it as the cluster center too. Then, the second most frequent token is clustered based on its similarity distance with the designated cluster center, according to the method discussed in Section 5.2. Also, to determine joining the existing cluster or forming a new one, we initialize the threshold to $\theta = 0.1$. This procedure continues until all tokens are clustered. In the update case, we apply the same method as SD-ClusPr. That is, upon uploading a document, the system decides to either perform re-clustering or updating existing clusters.

6. SECURITY ANALYSIS

The proposed clustering schemes are applicable in the context of searchable encryption and document retrieval systems. According to the three-tier architecture, described in Figure 1, client- and edge tiers are in the user premises, hence, the activities conducted and the user's key on these tiers are considered safe and trusted. The *Abstract* structures are kept on the edge tier in plain-text to enable us to measure the similarity with the search phrase and performing pruning.

On the other hand, activities performed on the cloud-tier are considered as dishonest and prone to different types of attacks. We are concerned about both internal (*i.e.,* affiliated parties) and external (*i.e.,* unaffiliated outside intruders) attackers who desire to learn the encrypted clustered tokens and documents. To explain the threats of the attackers, we provide the following preliminaries:

View: This term denotes the portion that is visible to the cloud during any given interaction among client, edge, and server. The central index and the set of clusters $C_1 ... C_n$, the trapdoor of the given encrypted search query $Q^{'}$, and the collection of encrypted documents $D^{'}$. In some models, $Q^{'}$ also contains a particular weight for each term. The search results related to $Q^{'}$ are considered as I_c. The view of expanded $Q^{'}$ and I_c are symbolized as $V(Q^{'})$ and $V(I_c)$ respectively.

Trace: This term denotes the information exposed about I_c. Our aim is to allow the attacker to infer the information of I_c as little as possible.

The View and Trace enclose all the information that the attacker would gain. To encrypt the document set we use probabilistic encryption model that is considered to be one of the most secure

encryption techniques [38, 56]. This does not utilize one-to-one mapping and so, $D^{'}$ is not prone to dictionary-based attacks [68]. Each token in a cluster is deterministically encrypted. Thus, each cluster in the View, only shows an encrypted mapping of the tokens and their co-occurrences in the plain-text format.

If any type of attacker can gain access to the cloud, he/she could only understand the importance of a particular encrypted token by observing the co-occurrences. It is technically possible to encrypt co-occurrences using homomorphic encryption [34] and perform computation on the co-occurrences while it is in the encrypted form. However, in Section 2, we discuss that this technique practically falls short on performance [51] and affects the real-time behavior of the search system. As such, in the current implementation, we use co-occurrence information in the plain-text format. Note that, even when the co-occurrences are not encrypted, the attacker cannot decrypt the token.

An attacker could obtain a Trace regarding $V(Q^{'})$. From that view, the attacker could only understand the importance of each search term from $Q^{'}$ by analyzing the associated weights of the query terms. Similar to the previous consideration, the attacker is not able to reveal the search terms from $Q^{'}$. In spite of a minimally trusted computing base, an attacker may still intend to access the system through man-in-the-middle, either *honest but compromised* or *untrusted* cloud providers to attack the confidentiality of the user data. By any means, if the attacker successfully performs a man-in-the-middle attack, he/she can access the document list $V(I_c)$ resulting from searching $Q^{'}$ with Trace. At this point, the attacker may only obtain the documents' names with encrypted contents that are unreadable.

There are methods (e.g., [35]) that can be used to tackle frequency attacks when the searches and cluster updates are predictable. Theoretically, an attacker could build a dictionary considering all the clusters' tokens by performing frequency attack. Eventually, the attacker tries to build a clone document set D' utilizing the dictionary. Although all of the tokens extracted from a particular document are sufficient to learn the topic of the document, it is not possible to unveil the whole document as we do not use all of the keywords of the document set to build the encrypted index. Besides, we encrypt the whole document at once instead of word level encryption before outsourcing it to the cloud. This procedure ensures that even if the document set is compromised on the cloud tier, it is impossible to perform a dictionary attack.

Even if the attacker knows the trace, he/she cannot understand what exactly the retrieved encrypted documents convey. Moreover, attacks can be occurred in the communication between the edge and cloud tiers. In this case, by monitoring the search process, an attacker could obtain the resultant document list for Q'. However, the attacker is not able to decrypt the documents, since they can be decrypted only when they are downloaded on the edge system.

An attacker could also attempt to modify data (*e.g.,* encrypted tokens and documents) in the clusters. Such attacks can potentially tamper with the integrity of user data. However, this type of attack could be detected, because neither the edge will be able to decrypt the modified tokens to form or update *Abstracts*, nor the user will be able to decrypt the retrieved documents in the original plain-text form. This is because of applying symmetric encryption (*e.g.,* AES encryption) on the user's data with keys managed by the user. Hence, in the event that the encrypted data are altered by an attacker, such data cannot be decrypted by the users' keys. Actually, protecting the user's key is crucial to restrain possible attacks. If the key is compromised, the system cannot detect the attacker and, therefore, both tokens and documents can be exposed.

7. PERFORMANCE EVALUATION

7.1. Experimental Setup

We developed a working version of ClusPr and made it available publicly in our Github*. We evaluate the performance of ClusPr using three distinct datasets that have different properties and

*https://git.io/fjDsq

volumes. We compare and analyze the clustering quality with other approaches that operate in encrypted or unencrypted domains. The experiments were conducted on a machine with two 10-core 2.8 GHz E5 Intel Xeon processors and 64 GB of memory.

To evaluate the performance of ClusPr in handling big data, we used a subset of `Amazon Common Crawl Corpus (ACCC)` dataset [4]. The whole dataset size $\approx$ 150 terabytes that contains different web-based contents, such as blogs and social media contents. We randomly selected $6,119$ documents that collectively form a $\approx$ 500 GB document set. The second dataset, named `Request For Comments (RFC)` [11], is domain- specific and includes documents about the internet and communication networks. RFC includes $2,000$ documents and its total size is $\approx$ 247 MB. The third dataset is `BBC` [8] that is not domain-specific and includes news in certain categories such as technology, politics, sports, entertainments, and business. It contains $2,225$ documents and is ≈ 5 MB. The reason for choosing this small dataset is that, unlike ACCC and RFC, each document of BBC is short and we can verify clusters' coherency manually. For each dataset, the documents are passed through Maui keyword extractor [48] to identify keywords semantically represent the document.

7.2. Evaluation Metrics and Baselines from Prior Works

For performance evaluation of ClusPr, we compare it against five other schemes, where two schemes cluster plain-text data and the other three schemes cluster encrypted data. Among the two, one of the schemes *W2V Kmeans*) is based on *K*-means clustering [24] where feature extraction is done based on Word2Vec [49] embedding.

Another scheme, *WordNet* [50], is an enhanced version of *K*-means that generates synonym set based on the input data and then, applies K-means clustering on the sets. Token distribution in WordNet is performed based on edge counting method, proposed by Wu and Palmer [50].

Three encrypted clustering schemes that have been used in the comparison are namely, S3BD [38], *HK*-means++ [66], and ClustCrypt [76]). We have discussed S3BD and *HK*-means++ in Section 2. *ClustCrypt* is the preliminary version of S-ClusPr. Their difference mainly lies in the way tokens are distributed across the clusters. In ClustCrypt, the relatedness is simply calculated based on contribution and co-occurrences metrics, whereas in S-ClusPr, the magnitude of both similarity and disparity are considered to measure the relatedness (see Section 4.3 for further details).

The goodness of clusters set can be quantified by a number of evaluation metrics. However, evaluating the performance of a clustering scheme is not as simple as counting errors in classification algorithm. Specifically, instead of considering the absolute values of cluster labels, cluster evaluation metrics either measure the separation of clustered data similar to ground truth set of classes or internal cluster validation. Internal cluster validation denotes that members belong to the same class should be more similar than members of other classes and vice versa. In practice, class label information is not always available in most of the application scenarios and, therefore, internal validation metrics are the only option for validation in such situation [57, 42].

As there is no ground truth for the considered datasets, we choose evaluation metrics that evaluate the clusters based on statistical analysis of the cluster members. We evaluate three widely-adopted clustering metrics, namely Silhouette coefficient (SC), Calinski-Harabasz index (CI), and Davies-Bouldin index (DI).

Silhouette Coefficient (SC) score interprets and validates intra-cluster consistency. In particular, the metric signifies how similar a cluster member is to its own cluster compared to the other clusters. The value of the SC score ranges from -1 to $+1$, where a high value indicates that a given member is well matched to its own cluster and poorly matched to the other ones. *Calinski-Harabasz Index (CI)* denotes how well-defined (*i.e.,* well-separated) the clusters are. The CI value of clusters is calculated based on the ratio of the sum of between-clusters dispersion to the sum of inter-cluster dispersion. A higher CI value indicates a more topically separated (*i.e.,* less overlapping) clustering and vice versa. Similar to the CI metric, *Davies Bouldin Index (DI)* is used to measure the goodness of separation across clusters and the reason we consider it in our evaluation is to verify the CI metric evaluation for the clusters. DI is calculated based on the ratio of within-cluster distances to the between-cluster distances. A lower DI value indicates a more topically-separated clustering and it is preferred. In

addition to these metrics, we measure the *clusters' coherency* to evaluate the quality of the topic-based clustering within each cluster. This is a similarity-based evaluation metric to calculate the average of all possible pair-wise token similarity for a given cluster. In fact, Coherency represents how the tokens in a cluster are related to a certain topic. Then, the average of coherency across all clusters is calculated to represent the overall quality of a certain clustering method.

We instrument the pre-trained Google News Word2vec model [49] to determine the similarity between any two given keywords. The model is a 300-dimension vector representation of three million phrases. The model requires a text dataset as input to build a vocabulary from the input dataset and learns vector representation of the words in the dataset. The model uses cosine similarity and provides the score ($-1 \leq similarity\ score \leq 1$) for any two given tokens. We note that, the pre-trained Word2vec model operates only on plain-text tokens. Subsequently, we do not encrypt the tokens while uploading for evaluation purposes. However, the proposed schemes assume tokens to be encrypted and do not use the properties of plain-text tokens.

7.3. Evaluation Results

7.3.1. Evaluating Silhouette Coefficient (SC) Score

Figure 3 shows the results of SC score evaluation on the three datasets and for varying number of clusters (in the horizontal axis). We note that, for this experiment, the value of K in W2V Kmeans, WordNet, and *HK*-means++ is randomly chosen and iteratively evolves. As such, we calculate the SC score for all the considered K values and show them in multiple data points in the figure. However, other schemes (namely, S-ClusPr, ClustCrypt, S3BD) are not iterative and provide only one SC score for their determined K values.

As the procedure of estimating the number of clusters is similar in ClustCrypt and S-ClusPr schemes, we can see that both of the schemes generate 69, 65, and 133 clusters for the `BBC`, `RFC`, and `ACCC` datasets, respectively. As `ACCC` is the largest and broadest (*i.e.,* not domain-specific) dataset, it yields the highest K value. `RFC` is not the smallest dataset, however, due to its domain-specific nature, it yields the lowest K value.

Figure 3 represents SC metric outcomes for S-ClusPrand the five other compared schemes. According to the figure, considering all of the datasets, overall top performers are WordNet and S-ClusPr. Moreover, S-ClusPr outperforms others in the `RFC` dataset. On the contrary, *blue HK*-means++ and S3BD underperform in most of the situation. The experiment indicates that the cluster sets generated by *HK*-means++ and S3BD contain less intra-cluster similarity. WordNet and S-ClusPr provide the highest intra-cluster similarity and hence, outperform others in all datasets.

7.3.2. Evaluating Calinski-Harabasz Index (CI)

Table IV represents CI metric outcomes for S-ClusPr and the five other schemes. According to the table, the RFC clusters provide large CI values compared to the BBC dataset, regardless of the employed clustering scheme. It is noteworthy that, we had the same observation for the ACCC dataset, however, we do not show its table due to the shortage of space. The superiority of RFC is because it is a domain-specific dataset with a few topics compared to the other two. Within Table IVb, we can see that although W2V-Kmeans significantly outperforms the other schemes for most of the K values, WordNet, ClustCrypt, and S-ClusPr also provide satisfactory CI values that imply well-partitioned clusters.

7.3.3. Evaluating Davies Bouldin Index (DI)

The DI values for the clusters, obtained by S-ClusPr and the compared schemes are expressed in Figure 4. In most of the scenarios, we observe that increasing the number of clusters reduces the DI value. This is because, typically, configuring clustering schemes to build more clusters on a given dataset leads to a higher coherency within each of the clusters.

According to the figure, we observe that WordNet scheme outperforms others. The DI value for S-ClusPr is in the acceptable range, which indicates that the scheme can offer a competitive goodness

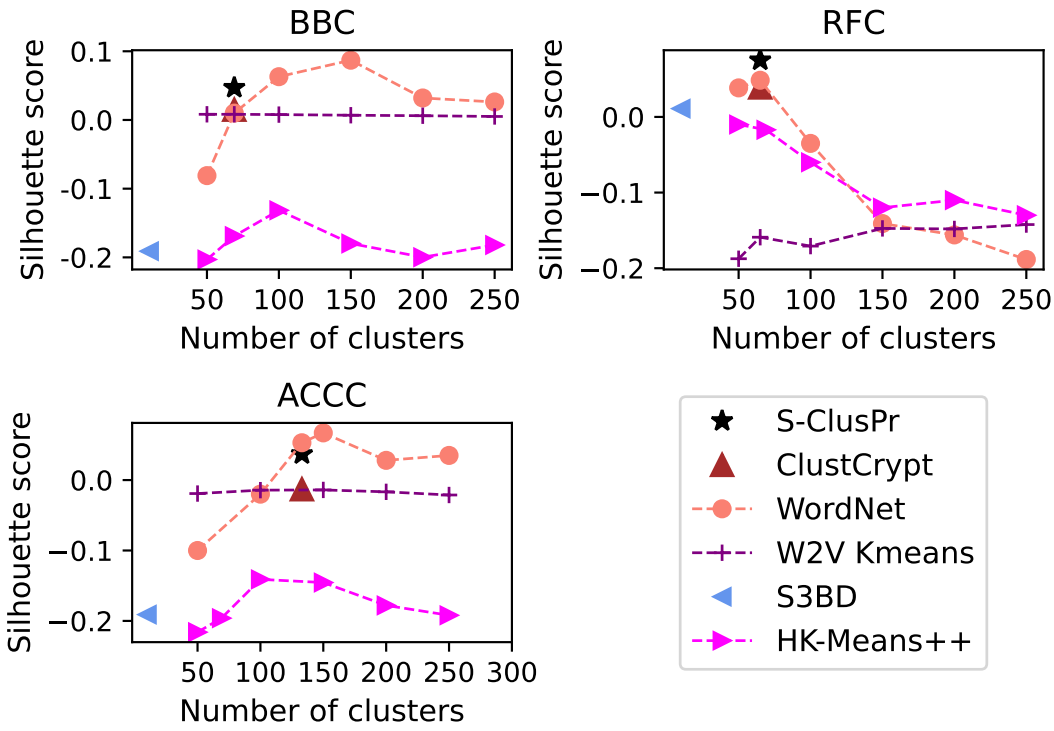


Figure 3. Silhouette Coefficient (SC) metric for each dataset. The results are obtained from S-ClusPr, *HK*-means++, ClustCrypt (that are encrypted-based clustering schemes), W2V-Kmeans, and WordNet clustering schemes (that operate on plain-text tokens).

of separation across clusters in compared to the most of other schemes. On the other hand, higher DI value yielded by *HK*-means++ signifies poor cluster separation.

7.3.4. Evaluating Cluster Coherency

Figure 5 shows the clusters' coherency on the three datasets using various clustering schemes. Using S-ClusPr, 69, 65, and 133 clusters are created for the `BBC`, `RFC`, and `ACCC` datasets, respectively. As `ACCC` is the largest and broadest (*i.e.,* not domain-specific) dataset, it yields the highest K value. `RFC` is not the smallest dataset, however, due to its domain-specific nature, it yields the lowest K value. For the same reason, across the three datasets, S-ClusPr offers the highest coherency value (≈ 0.16) for the `RFC` dataset.

In compare to ClustCrypt, we notice that S-ClusPr offers a negligible coherency improvement ($\approx 6\%$) for the `BBC` and `RFC` datasets. However, for the `ACCC` dataset, S-ClusPr improves the coherency by approximately 31%.

Analysis of the plain-text-based schemes reveal that, WordNet clusters offer the highest coherency value. This is expected, because it is difficult for an encrypted clustering scheme (*e.g.,* S-ClusPr) to outperform the unencrypted ones, since they do not have access to the semantics of the tokens [50] to build the clusters. However, we observe that the coherency offered by S-ClusPr competes with the one offered by the K-means scheme. In particular, S-ClusPr provides a higher coherency value than K-means for the `RFC` and `BBC` datasets.

To evaluate the suitability of estimated number of clusters (K) by S-ClusPr, we configure both K-means and WordNet to use the estimated K number of clusters for the studied datasets. According to the figure, for `RFC` and `BBC`, S-ClusPr suggested sets of K clusters offer a higher coherency than K-means and a comparable one to WordNet. In the case of `ACCC`, S-ClusPr even outperforms WordNet in terms of coherency.

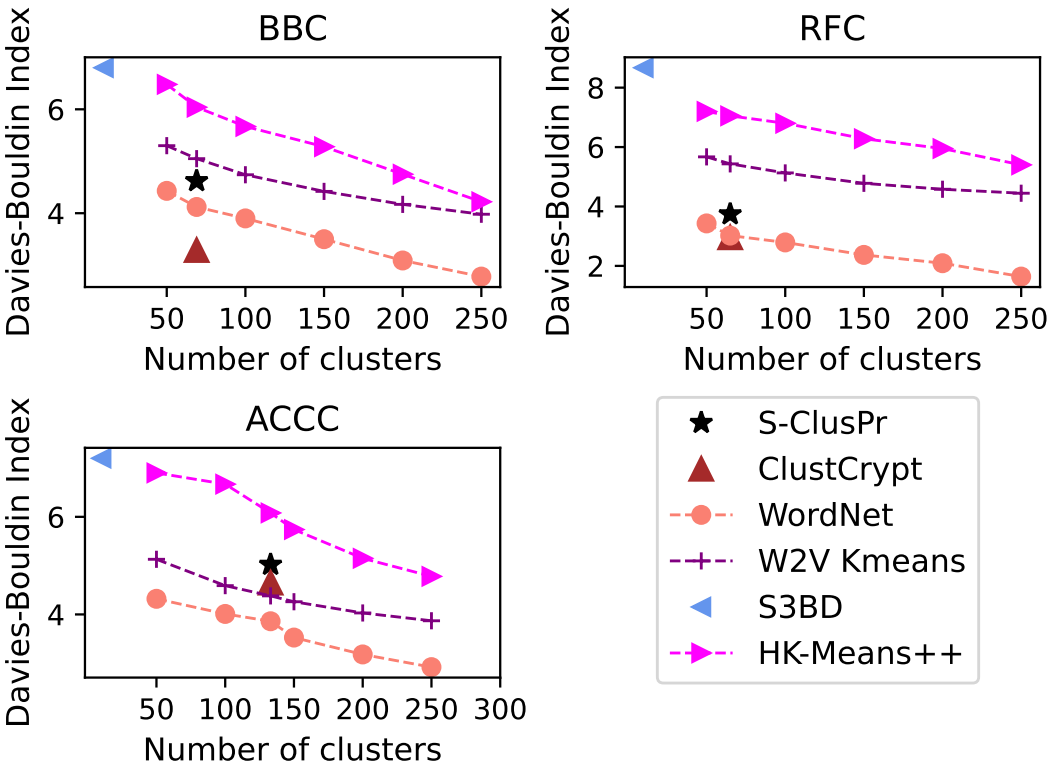


Figure 4. Davies-Bouldin Index (DI) for each dataset using different clustering schemes.

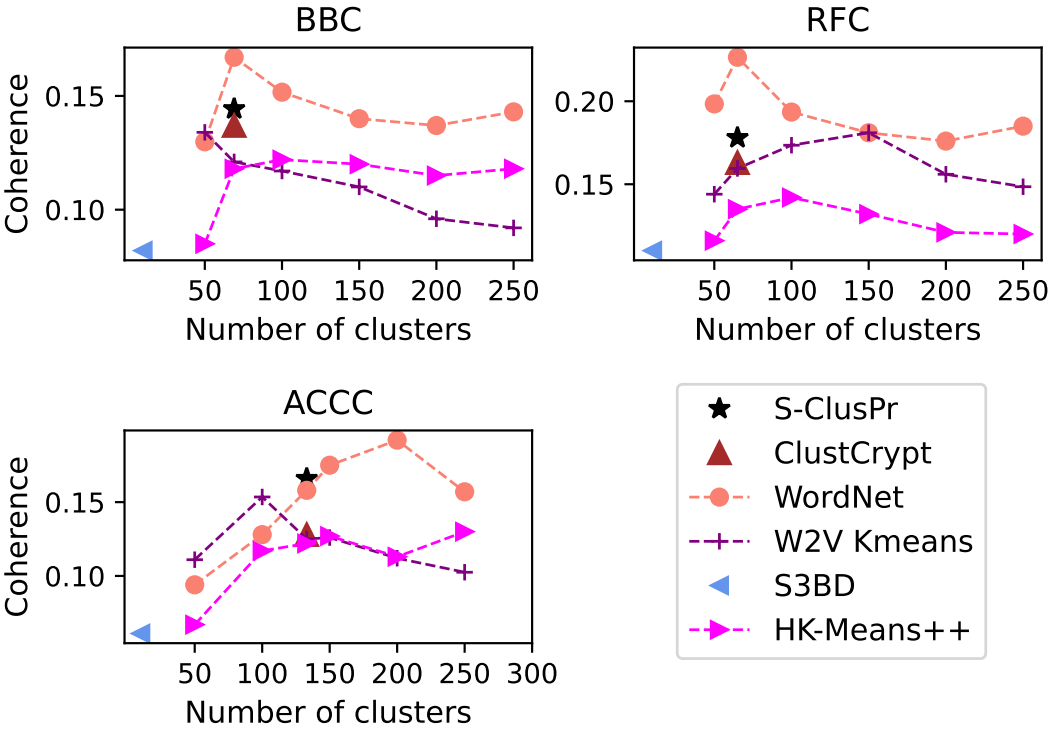


Figure 5. Cluster coherency for each dataset.

(a) BBC

No. of Cluster	Approaches					
	HK-means++	WordNet	W2V Kmeans	S3BD	ClustCrypt	S-ClusPr
10	-	-	-	8.7	-	-
50	25.43	**277.53**	11.16	-	-	-
69	18.47	**253.60**	9.22	-	11.70	13.58
100	11.13	**203.87**	7.37	-	-	-
150	14.05	**164.43**	5.81	-	-	-
200	10.17	122.51	4.93	-	-	-
250	12.02	97.15	4.38	-	-	-

(b) RFC

No. of Cluster	Approaches					
	HK-means++	WordNet	W2V Kmeans	S3BD	ClustCrypt	S-ClusPr
10	-	-	-	1247.20	-	-
50	1730.26	4320.63	**60380.05**	-	-	-
65	1945.42	3980.75	**51564.61**	-	23760.64	**29439.30**
100	1834.64	3660.78	**24374.17**	-	-	-
150	1684.47	3110.25	**18684.33**	-	-	-
200	846.71	2572.89	16746.74	-	-	-
250	436.43	1834.58	15139.11	-	-	-

Table IV. Calinski-Harabasz Index for the datasets.

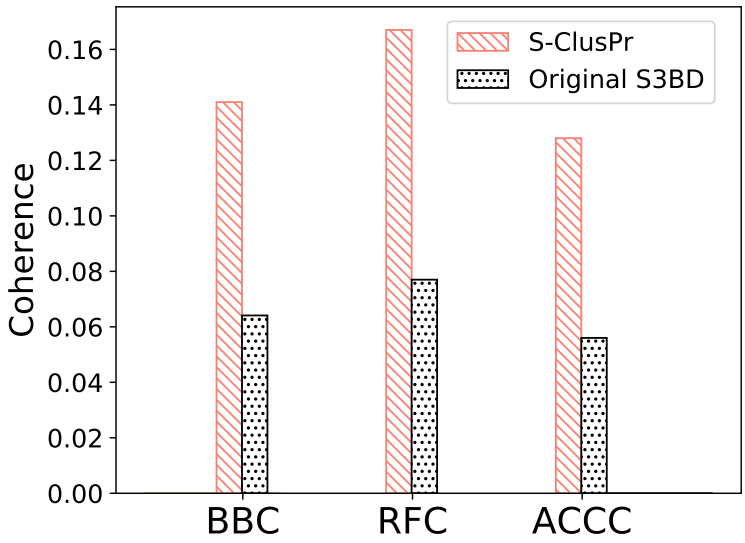


Figure 6. Comparing the impact of clustering using S-ClusPr against original clustering of S3BD for the studied datasets.

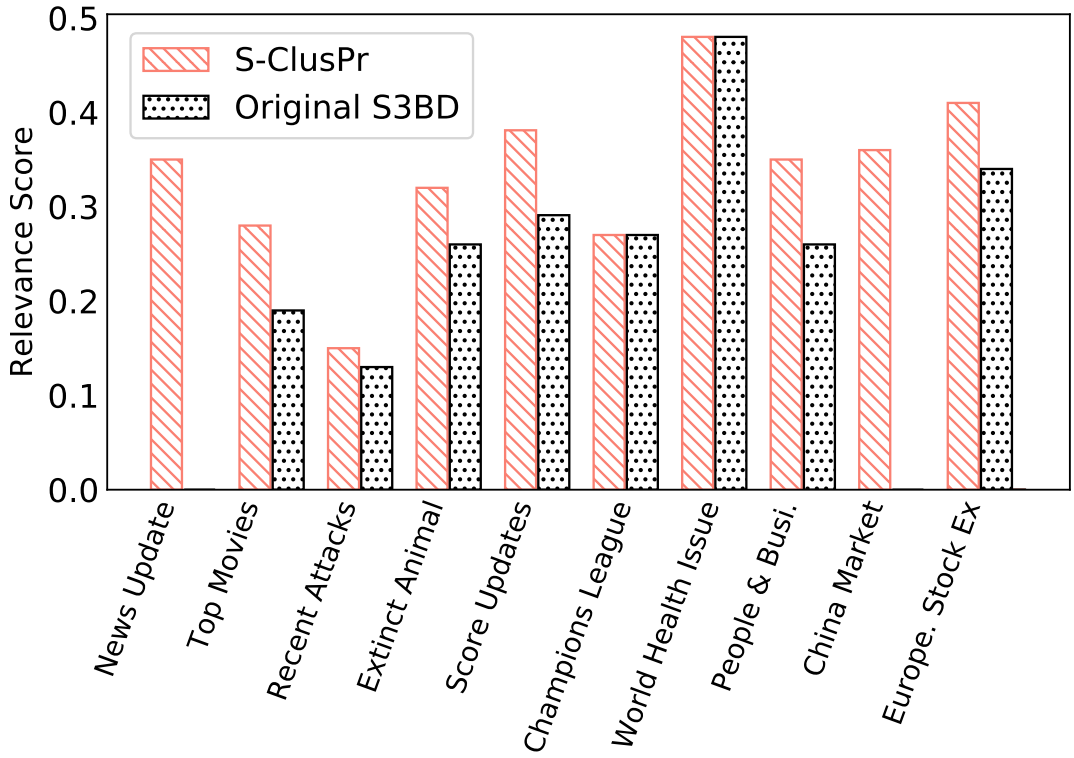


Figure 7. Comparing the relevancy of search results using S-ClusPr vs original S3BD clustering in BBC dataset. The value of relevancy is calculated based on TSAP@10 scoring metric.

7.3.5. Analyzing the Impact of S-ClusPr on Searchable Encryption Systems One objective of this research is to enhance the performance of S3BD secure search system. As such, we instrumented S-ClusPr in S3BD and compared the coherency of resulting clusters with its original clustering scheme that predetermines a value for $k = 10$. Moreover, its center selection only considers the co-occurrences. In this experiment, we intend to evaluate the improvement that S-ClusPr achieves within S3BD on the three studied datasets. In this experiment, the estimated values of K for `BBC`, `RFC`, and `ACCC` are 69, 65, and 133, respectively.

a) Impact on the Clustering Coherency of S3BD. Figure 6 shows that for all the studied datasets, clusters generated by S-ClusPr have remarkably higher coherency than the original clustering scheme of S3BD. This shows determining number of clusters based on dataset characteristics and choosing center tokens based on the centrality concept is effective. Our hypothesis is that, such efficiency improves the accuracy and offers more relevant semantic search results. This is because tokens of the clusters are more congruent to the clusters' topics, hence, more effective pruning is accomplished. For further evaluation of this hypothesis, next experiments concentrate on the impact of S-ClusPr on the search quality.

b) Impact on the Search Accuracy of S3BD The purpose of improving the clusters' coherency in this study is to ultimately enhance the search accuracy by retrieving more relevant documents. To evaluate the impact of such improvement, in this part, we compare and analyze how the search accuracy of S3BD system is affected by utilizing S-ClusPr's clusters against the circumstance where

ACCC Dataset	BBC Dataset	RFC Dataset
Orlando Magic	News Update	Internet
Samsung Galaxy	Top Movies	TCP
Baseball routine	Recent Attacks	Fiber Doctor
Recommendation	Endangered Animals	Wifi
North America	Score Updates	IoT
Tennis Tournament	Champions League	Radio Frequency
Holy Martyr	World Health Issue	UDP
Library	People and Business	Edge Computing
Stardock	China Market	Encryption Schemes
Orthodox Church	European Stock Exchange	Broadcasting

Table V. Benchmark queries for each one of the studied datasets.

its original clustering method is utilized.For the evaluation, we generated a set of 10 benchmark search queries that are listed in Table V.

To measure the relevancy of search results for each query, we use *TREC-Style Average Precision* scoring method [46]. This method works based on the recall-precision concept and the score is calculated by $\sum_{i=0}^{N} r_i/N$, where r_i denotes the score for i^{th} retrieved document and N is the cutoff number (number of elements in the search results) that we consider as 10. Therefore, we call it *TSAP@10*.

We measure TSAP@10 score only for the RFC dataset and its benchmark queries. The reason is that it is domain-specific and feasible to determine the relevancy of the retrieved documents. To compare the relevancy provided by S-ClusPr against the original S3BD clustering, we apply the benchmark queries to the S3BD search system. In Figure 7, the relevancy score of the results for each query when the two clustering schemes are applied are measured and presented. According to the Figure, for most of the queries, S-ClusPr clustering offers a higher relevancy score. For the two queries that have identical *TSAP@10* score, their retrieved document lists are equivalent. Also, S-ClusPr clusters provide score for *News Update* and *China Market* benchmark queries, whereas original S3BD clusters do not retrieve any relevant documents for these queries.

c) Impact on the Search Time of S3BD Figure 8 presents the total search time of the benchmark queries for each dataset. The search time is measured as the turnaround time of searching each query—from the time a query is issued until the result set is received. To eliminate the impact of any randomness in the computing system, we searched each set of benchmarks 10 times and reported the results in form of box plots. The figure indicates that when S-ClusPr clustering is utilized, the search time is significantly shorter than the circumstance where the original S3BD clustering is used. Longer search time impacts the scalability and real-time quality of the search operation on big data. Analyzing Figures 6 to 8 reveals that integrating S-ClusPr in the search system, not only makes it more accurate, but makes it faster and more scalable too.

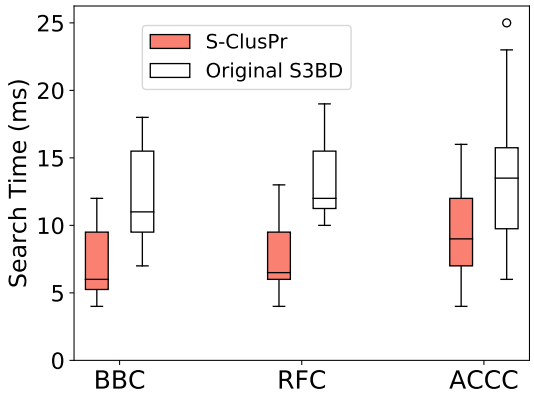
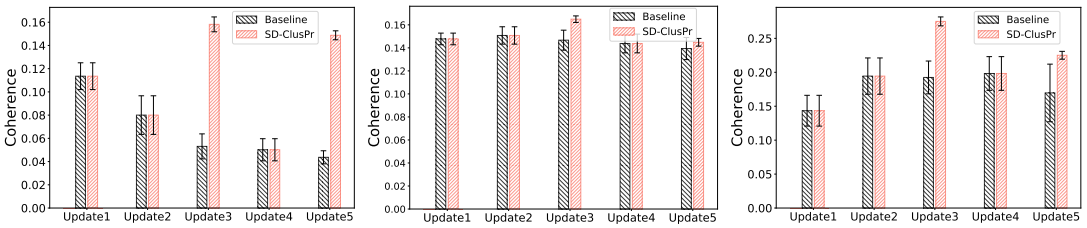


Figure 8. Search time of S3BD when S-ClusPr is used for clustering versus when the original S3BD clustering is used.



(a) BBC Dataset (b) RFC Dataset (c) ACCC Dataset

Figure 9. Clusters' coherency for different updates of the three studied datasets when SD-ClusPr is applied with and without re-clustering option.

7.3.6. Evaluation of Clustering Coherency for Dynamic Schemes In this part, we analyze the effectiveness of dynamic clustering schemes (SD-ClusPr and FD-ClusPr). We mention in Section 5.3 that FD-ClusPr is a specific case of SD-ClusPr. Hence, we only consider the SD-ClusPr scheme for evaluation. To this end, we leverage the three studied datasets and build subsets that each one serves as a batch update. Specifically, we consider an existing set of clusters based on 500 documents for each dataset. Then, we sample five times to create a list of five updates that each one includes a set of documents. List U includes the pairs of update names and the size of each update as follows: $U =< (U_1, 25), (U_2, 50), (U_3, 100), (U_4, 20), (U_5, 200) >$. To assure that the results are not biased to any particular sample, we performed the sampling procedure 10 independent times and report the mean and 95% confidence interval of the analysis in the results. The reason we designated U_3 and U_5 to be larger is to examine SD-ClusPr decision in re-clustering. To evaluate the scheme in terms of the cluster coherency, we build a *baseline* version from SD-ClusPr that does not consider re-clustering. The baseline only performs clustering based on existing clusters (as explained in Algorithm 2) to accommodate the new updates.

Figures 9a, 9b, and 9c, respectively, present cluster coherency of five different batch updates of `BBC`, `RFC`, and `ACCC` respectively applying SD-ClusPr scheme. In Figure 9a, we observe that the coherency of clusters are decreased in baseline for U_3 whereas the coherency obtained for SD-ClusPr beats the previous by around 105%. We observe the similar pattern of coherency variation for U_5. For baseline, the lowest coherency is obtained in U_5. On the contrary, in SD-ClusPr, we observe around 115% improvement in coherency for U_5.

According to Figure 5, clusters formed for the `RFC` dataset shows the highest coherency. Similarly, in 9b, we observe the highest coherency for all updates in compare with other datasets. With respect to baseline, we observe that SD-ClusPr causes minor improvements in coherency of both U_3 and U_5. Since the documents are more domain-specific, clusters do not lose coherency

significantly from one update to the other. As such, we do not observe significant improvements by SD-ClusPr. Similar to BBC and RFC, in Figure 9c, we observe improvement in the coherency for ACCC dataset. In particular, the improvement in coherency for U_3 and U_5 is approximately 45% and 35%, respectively.

From these experiments, we conclude that ClusPr scheme can improve the coherency of clustering even for dynamic datasets. Specifically, we observed that for sufficiently large batches, such as U_3 and U_5, SD-ClusPr decides to re-cluster that remarkably improves the clustering coherency.

8. CONCLUSIONS

In this research, we developed ClusPr as a solution for topic-based clustering of both static and dynamic unstructured encrypted big datasets. ClusPr approximates the number of clusters for a dataset within a feasible time complexity. For that purpose, ClusPr leverages the tokens' co-occurrences to measures the tendency of each token to stay with or segregate from other tokens and use that to estimate the number of clusters. Next, we develop a probabilistic approach to determine the center of each cluster and disseminate encrypted tokens to the most topically related cluster. Experimental evaluations reveal that our clustering scheme for static datasets (S-ClusPr) can improve the clustering coherency on average by 65%. Similarly, the scheme for semi-dynamic and dynamic datasets (SD-ClusPr) can improve the coherency by 55%. By incorporating ClusPr within the context of a secure semantic search system, we learned that the more coherent and accurate topic-based clustering can improve the relevancy of search results. Although ClusPr outperforms other encryption-based clustering solutions, it is difficult for it to beat the schemes that operates on plain-text tokens. There are several avenues to extend this research work. One avenue is to identify the user search interest and leverage that to create more representative abstracts. Another avenue is to improve the cluster pruning component to reduce the search time operation.

REFERENCES

1. Amazon Kendra . https://aws.amazon.com/kendra/, Accessed April '20.
2. Cloud leak: How a verizon partner exposed millions of customer accounts. https://www.upguard.com/breaches/verizon-cloud-leak, Accessed April '20.
3. Consensus Clustering. https://towardsdatascience.com/consensus-clustering-f5d25c98eaf2, Accessed April '20.
4. Common Crawl on Amazon Web Services (AWS). https://aws.amazon.com/public-datasets/common-crawl/, Accessed February 2, 2020.
5. Every single yahoo account was hacked - 3 billion in all. https://www.money.cnn.com/2017/10/03/technology/business/yahoo-breach-3-billion-accounts/index.html, Accessed February '21.
6. Clustering metrics better than the elbow-method. https://towardsdatascience.com/clustering-metrics-better-than-the-elbow-method-6926e1f723a6, Accessed July 8, 2021.
7. Statistics Solution. https://www.statisticssolutions.com/using-chi-square-statistic-in-research/, Accessed July 8, 2021.
8. BBC news classification. https://www.kaggle.com/c/learn-ai-bbc, Accessed March '20.
9. Consuming Streaming Data. https://developer.twitter.com/en/docs/tutorials\/consuming-streaming-data, Accessed March '20.
10. Criminal Records. https://staterecords.org/criminal.php, Accessed March '20.
11. RFC (request for comments) series. https://old.datahub.io/dataset/rfcs, Accessed March '20.
12. Twitter stream api dataset. https://github.com/shreybatra/Twitter-Stream-API-Dataset, Accessed March '20.
13. 2000 HUB5 english evaluation transcripts. https://catalog.ldc.upenn.edu/LDC2002T43, Accessed March '21.
14. The Museum of Modern Art Data. https://tapoueh.org/blog/2018/07/batch-updates-and-concurrency, Accessed March '21.
15. Clustering algorithms: K-means. https://www.cs.princeton.edu/courses/archive/spr08/cos435/Class$_{n}$otes/clustering2$_{t}$oPost.pdf, Accessed November, 2020.
16. DBSCAN:what is it? when to use it? how to use it? https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818, Accessed November, 2020.
17. Ecb vesus cbc mode aes encryption. https://datalocker.com/what-is-the-difference-between-ecb-mode-versus-cbc-mode-aes-encryption/, Accessed November, 2020.

18. Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the 8th International conference on database theory*, ICDT'01, pages 420–434, January 2001.
19. Swati Aggarwal, Nitika Agarwal, and Monal Jain. Performance analysis of uncertain k-means clustering algorithm using different distance metrics. In *Computational Intelligence: Theories, Applications and Future Directions-Volume I*, pages 237–245. Springer, 2019.
20. Sahan Ahmad, SM Zobaed, Raju Gottumukkala, and Mohsen Amini Salehi. Edge computing for user-centric secure search on cloud-based encrypted big data. In *Proceedings of the 21st International Conference on High Performance Computing and Communications (HPCC'19)*, pages 662–669, August 2019.
21. Jalal Al-Muhtadi, Basit Shahzad, Kashif Saleem, Wasif Jameel, and Mehmet A Orgun. Cybersecurity and privacy issues for socially integrated mobile healthcare applications operating in a multi-cloud environment. *Jounral of Health informatics journal*, 25(2):315–329, May 2019.
22. L. A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2):22–28, March 2003.
23. Gema Bello-Orgaz, Jason J Jung, and David Camacho. Social big data: Recent achievements and new challenges. *Journal of Information Fusion*, 28:45–59, March 2016.
24. Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.
25. Kenneth P Burnham and David R Anderson. Kullback-leibler information as a basis for strong inference in ecological studies. *Journal of Wildlife research*, 28(2):111–119, 2001.
26. Fazli Can and Esen A. Ozkarahan. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *Journal of ACM Trans. Database Syst.*, 15(4):483–517, December 1990.
27. Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*, pages 561–580. Springer, 2012.
28. Baojiang Cui, Zheli Liu, and Lingyu Wang. Key-aggregate searchable encryption (kase) for group data sharing via cloud storage. *Transactions of Computers*, 65(8):2374–2385, 2016.
29. Douglass R Cutting, David R Karger, Jan O Pedersen, and John W Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *ACM SIGIR Forum*, volume 51, pages 148–159, August 2017.
30. Paul Inuwa Dalatu. Time complexity of k-means and k-medians clustering algorithms in outliers detection. *Journal of Pure and Applied Mathematics*, 12(5):4405–4418, November 2016.
31. W. Diffie and M. Hellman. New directions in cryptography. *Transactions on Information Theory*, 22(6):644–654, November 1976.
32. Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology*, EUROCRYPT'15, pages 617–640, April 2015.
33. Wei Fu and Patrick O Perry. Estimating the number of clusters using cross-validation. *Journal of Computational and Graphical Statistics*, pages 1–12, January 2019.
34. K. Gai and M. Qiu. Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers. *Transactions on Industrial Informatics*, 14(8):3590–3598, August 2018.
35. Marilyn George, Seny Kamara, and Tarik Moataz. Structured encryption and dynamic leakage suppression. In *Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 370–396, October 2021.
36. P. Hammersley. Editorial – information and information systems. *The Computer Journal*, 32(3), January 1989.
37. Chenyue W Hu, Steven M Kornblau, John H Slater, and Amina A Qutub. Progeny clustering: a method to identify biological phenotypes. *Journal of Scientific reports*, 5:12894, August 2015.
38. Mohsen Amini Salehi Jason Woodworth. S3BD: secure semantic search over encrypted big data in the cloud. *Journal of Concurrency and Computation:Practice and Experience (CCPE)*, 28(11), December 2018.
39. Se-Hoon Jung and Jong-Chan Kim. Efficiency improvement of classification model based on altered k-means using pca and outlier. *Journal of Software Engineering and Knowledge Engineering*, 29(05):693–713, May 2019.
40. Nesrine Kaaniche and Maryline Laurent. A secure client side deduplication scheme in cloud storage environments. In *Proceedings of the 6th International Conference on New Technologies, Mobility and Security*, NTMS'14, pages 1–7, March 2014.
41. Vinod Kumar, Rajendra Kumar, Santosh Kumar Pandey, and Mansaf Alam. Fully homomorphic encryption scheme with probabilistic encryption based on euler's theorem and application in cloud computing. In *Big Data Analytics*, pages 605–611. Springer, 2018.
42. Bum Chul Kwon, Ben Eysenbach, Janu Verma, Kenney Ng, Christopher De Filippi, Walter F Stewart, and Adam Perer. Clustervision: Visual supervision of unsupervised clustering. *IEEE transactions on visualization and computer graphics*, 24(1):142–151, 2017.
43. Tilman Lange, Volker Roth, Mikio L Braun, and Joachim M Buhmann. Stability-based validation of clustering solutions. *Journal of Neural computation*, 16(6):1299–1323, June 2004.
44. Ping Li, Jin Li, Zhengan Huang, Chong-Zhi Gao, Wen-Bin Chen, and Kai Chen. Privacy-preserving outsourced classification in cloud computing. *Journal of Cluster Computing*, 21(1):277–286, March 2018.
45. Xiaoyong Liu and W. Bruce Croft. Cluster-based retrieval using language models. In *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 186–193, July 2004.
46. A. K. Mariappan, R. M. Suresh, and V. Subbiah Bharathi. A comparative study on the effectiveness of semantic search engine over keyword search engine using tsap measure. *Journal of Computer Applications EGovernance and Cloud Computing Services*, pages 4–6, December 2012.
47. Angel Latha Mary and KR Shankar Kumar. A density based dynamic data clustering algorithm based on incremental dataset. *Journal of Computer Science*, 8(5):656–664, February 2012.
48. Olena Medelyan, Eibe Frank, and Ian H. Witten. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 14th Conference on Empirical Methods in Natural Language*, EMNLP '09, pages 1318–1327, August 2009.
49. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

50. George A Miller. Wordnet: a lexical database for english. *Journal of Communications of the ACM*, 38(11):39–41, November 1995.
51. Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, October 2011.
52. Hongping Pang and Baocang Wang. Privacy-preserving association rule mining using homomorphic encryption in a multikey environment. *Systems Journal*, 15(2):3131–3141, 2020.
53. John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1855–1870, May 2015.
54. Dan Pelleg and Andrew W Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning*, ICML'00, pages 727–734, June 2000.
55. Hoang Pham, Jason Woodworth, and Mohsen Amini Salehi. Survey on secure search over encrypted data on the cloud. *Journal of Concurrency and Computation: Practice and Experience*, 31(17):e5284, 2019.
56. Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100, October.
57. Mayra Z Rodriguez, Cesar H Comin, Dalcimar Casanova, Odemir M Bruno, Diego R Amancio, Luciano da F Costa, and Francisco A Rodrigues. Clustering algorithms: A comparative approach. *Journal of PloS one*, 14(1):e0210236, 2019.
58. Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, November 1987.
59. Mohsen Amini Salehi, Thomas Caldwell, Alejandro Fernandez, Emmanuel Mickiewicz, Eric WD Rozier, Saman Zonouz, and David Redberg. RESeED: a secure regular-expression search tool for storage clouds. *Journal of Software: Practice and Experience*, 47(9):1221–1241, September 2017.
60. Wenhai Sun, Wenjing Lou, Y Thomas Hou, and Hui Li. Privacy-preserving keyword search over encrypted data in cloud computing. In *Secure cloud computing*, pages 189–212. Springer, 2014.
61. Wenhai Sun, Bing Wang, Ning Cao, Ming Li, Wenjing Lou, Y Thomas Hou, and Hui Li. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In *Proceedings of the 8th symposium on Information, computer and communications security*, SIGSAC'13, pages 71–82, May 2013.
62. Thaddeus Tarpey. Linear transformations and the k-means clustering algorithm: applications to clustering curves. *Journal of the american statistician*, 61(1):34–40, 2007.
63. Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, January 2002.
64. Thanh N Tran, Klaudia Drab, and Michal Daszykowski. Revised dbscan algorithm to cluster data with dense adjacent clusters. *Journal of Chemometrics and Intelligent Laboratory Systems*, 120:92–96, January 2013.
65. Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the 9th international conference on Knowledge discovery and data mining*, pages 206–215, 2003.
66. Chen Wang, Andi Wang, Xinyu Liu, and Jian Xu. Research on k-means clustering algorithm over encrypted data. In *Proceedings of International Symposium on Cyberspace Safety and Security*, pages 182–191, December 2019.
67. Ding Wang, Kaan Ozbay, and Zilin Bian. A mixture model-based clustering method for fundamental diagram calibration applied in large network simulation. In *Proceedings of 23rd International Conference on Intelligent Transportation Systems*, ITSC'23, pages 1–6, 2020.
68. Ding Wang and Ping Wang. Offline dictionary attack on password authentication schemes using smart cards. In *Information Security*, pages 221–237. Springer International Publishing, 2015.
69. Jason Woodworth, Mohsen Amini Salehi, and Vijay Raghavan. S3C: An architecture for space-efficient semantic search over encrypted data in the cloud. In *Proceedings of the 4th IEEE International Conference on Big Data*, Big Data'16, pages 3722–3731, December 2016.
70. Kai Xing, Chunqiang Hu, Jiguo Yu, Xiuzhen Cheng, and Fengjuan Zhang. Mutual privacy preserving k-means clustering in social participatory sensing. *Transactions on Industrial Informatics*, 13(4):2066–2076, 2017.
71. Jinxi Xu and W. Bruce Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd International ACM Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 254–261, August 1999.
72. Ke Zhang, Jiahuan Long, Xiaofen Wang, Hong-Ning Dai, Kaitai Liang, and Muhammad Imran. Lightweight searchable encryption protocol for industrial internet of things. *Transactions on Industrial Informatics*, 17(6):4248–4259, 2020.
73. Mingwu Zhang, Yu Chen, and Jiajun Huang. Se-ppfm: A searchable encryption scheme supporting privacy-preserving fuzzy multikeyword in cloud systems. *Journal of Systems*, 15(2):2980–2988, 2020.
74. Lu Zhou, Youwen Zhu, and Aniello Castiglione. Efficient k-nn query over encrypted data in cloud with limited key-disclosure and offline data owner. *Computers & Security*, 69:84–96, 2017.
75. Youwen Zhu and Xingxin Li. Privacy-preserving k-means clustering with local synchronization in peer-to-peer networks. *Journal of Peer-to-Peer Networking and Applications*, 13(6), 2020.
76. SM Zobaed, Sahan Ahmad, Raju Gottumukkala, and Mohsen Amini Salehi. Clustcrypt: Privacy-preserving clustering of unstructured big data in the cloud. In *Proceedings of the 21st International Conference on High Performance Computing and Communications*, HPCC'19, pages 609–616. IEEE, August 2019.
77. Sm Zobaed and Mohsen Amini Salehi. Big data in the cloud. In Laurie A. Schintler and Connie L. McNeely, editors, *Encyclopedia of Big Data*. Springer, 2018.