ClustCrypt: Privacy-Preserving Clustering of Unstructured Big Data in the Cloud

SM Zobaed*, Sahan Ahmad*, Raju Gottumukkala†, and Mohsen Amini Salehi*

*School of Computing & Informatics
†Informatics Research Institute
University of Louisiana at Lafayette, Louisiana 70504, USA
Email: {sm.zobaed1, sahan.ahmad1, raju, amini}@louisiana.edu

Abstract—Security and confidentiality of big data stored in the cloud are important concerns for many organizations to adopt cloud services. One common approach to address the concerns is client-side encryption where data is encrypted on the client machine before being stored in the cloud. Having encrypted data in the cloud, however, limits the ability of data clustering, which is a crucial part of many data analytics applications, such as search systems. To overcome the limitation, in this paper, we present an approach named ClustCrypt for efficient topic-based clustering of encrypted unstructured big data in the cloud. ClustCrypt dynamically estimates the optimal number of clusters based on the statistical characteristics of encrypted data. It also provides clustering approach for encrypted data. We deploy ClustCrypt within the context of a secure cloud-based semantic search system (S3BD). Experimental results obtained from evaluating ClustCrypt on three datasets demonstrate on average 60% improvement on clusters' coherency. ClustCrypt also decreases the search-time overhead by up to 78% and increases the accuracy of search results by up to 35%.

Index Terms—Clustering; big data; privacy; unstructured data; cloud services;

I. INTRODUCTION

Many organizations own high volume of unstructured documents in forms of reports, emails, or web pages. The size of these data is expected to reach 44 zetabytes within the next two years [7]. Cloud providers offer scalable and convenient services to store, process, and analyze the massive volume of data, which is also known as big data. As such, organizations use cloud services to relieve from the burden of storing large volume of data locally. However, the outsourced contents (documents) often contain private or sensitive information (*e.g.,* criminal or financial reports [38]) that need to be protected against internal and external cloud attackers. In fact, data security and privacy concerns have made many organizations reluctant to use cloud. For instance, in 2018, more than 14 million Verizon customers' accounts information were leaked from their cloud repository [3]. In another instance, confidential information of more than three billion Yahoo users were exposed [2]. In [6], numerous similar data privacy violation incidents are reported. These incidents vividly highlight the importance of this issue in the cloud era.

An ideal privacy preserving cloud solution is expected to enable organizations to securely utilize cloud services, while providing access and search ability only to authorized users. Such a solution should be lightweight and users can have it on their thin-clients (*e.g.,* smartphones) with storage, energy, and data processing constraints. Client-side encryption [35], in which documents are encrypted with private keys before outsourcing them to the cloud, is a promising approach to achieve data privacy. However, the ability to process and search the encrypted documents is lost.

Searchable Encryption systems (*e.g.,* [15], [31]) have been developed to enable privacy preserving search ability over encrypted data. Such systems predominantly extract keywords (aka tokens) from documents and leverage them to carry out the search operation. The extracted keywords and the documents they appeared in are mapped to an index structure [15], [35], which is then traversed against a search query to find relevant documents. A problem arises for big datasets where the index becomes the processing bottleneck to conduct real-time search operation [21]. For instance, in S3C [35] search tool, for a 100-petabyte dataset the index size grows to ≈ 300 GB. Traversing such a large index affects searching timeliness and can be potentially impacted by hardware limitations [32]. To resolve the bottleneck, the index structure can be partitioned to several *clusters* [12], [36]. Then, for a given search query, the search space is pruned and limited to the clusters that are relevant to the query.

Although numerous data clustering methods exist, they are not appropriate for encrypted big data because of the following challenges: *First,* in the encrypted domain the original data is not available. Therefore, prior works (*e.g.,* [12], [36], [37]) suggest making use of statistical characteristics as the clustering metric for encrypted data. For instance, in S3BD [21], which is a search system for encrypted big data, keywords' co-occurrences in a document set is used to cluster keywords. However, in S3BD, a constant number of clusters (k) is considered, regardless of the dataset characteristics. *Second,* complexity of the clustering methods (*e.g.,* K-means [13]) make them unscalable for big data [8].

To address these two challenges, in this research, we investigate *how to optimally and scalably cluster keywords in encrypted big datasets?*. The outcome of this research is ClustCrypt that enhances clustering of encrypted keywords by estimating the appropriate number of clusters (k) and distributing keywords across them. Unlike traditional k-means-based clustering approaches, where (k) is chosen arbitrarily [13], ClustCrypt estimates k by probabilistically inferring the underlying semantic similarity among the encrypted tokens. The probabilistic calculation measures the tendency for each token to be separate from others. It is noteworthy that ClustCrypt has the advantage of improving the search quality and timeliness without implying any architectural changes to the existing systems. We develop ClustCrypt and evaluate it on three different datasets. We compare and analyze the the number and coherency of resulting clusters against baseline and state-

of-the-art clustering approaches that operate on plain-text data.

In summary, contributions of this paper are as follows:

- Estimating the appropriate number of clusters (k) for a given encrypted dataset based on its characteristics.
- Proposing a method to distribute encrypted keywords to relevant clusters.
- Evaluating clusters in terms of coherency and their impact on the search results.

The rest of the paper is organized as follows. In Sections II and III, we discuss related works and background. We explain ClustCrypt in Section IV. Then, in Section V, we discuss the threat model and provide security analysis of ClustCrypt. Sections VI and VII explain results and conclusion.

II. RELATED WORKS

Multiple methods have been provided to estimate the suitable number of clusters (k), and clustering processes. Determining the optimum number of clusters k is an NP-hard problem [29]. Hence, research has been undertaken to provide heuristic methods in which k is approximated and clustering approaches. In this section, we review such prior studies and position our work corresponding to them.

Many clustering approaches are independent from the dataset they are applied on. Such generic clustering methods, such as k-means, populate a predefined set of k clusters based on the convergence of center shifting [22], [30]. For a dataset with n data points, k clusters, I iterations, and m attributes, the time complexity of k-means is $O(n \times k \times I \times m)$, which is not scalable for big data [1], [33]. In contrast, after determining k, ClustCrypt populates clusters by achieving the two following steps. First, from the set of data points, we find appropriate centers for each cluster and assign each token to proper cluster. The overall time complexity of the two steps is $O(n^2)$.

Pelleg and Moore [30] proposed a framework, named x-means clustering, to approximate k via learning methods. The framework is a modified version of k-means clustering that starts with one cluster (*i.e.,* k=1) and iteratively subdivides it maintaining stopping criterion [19]. An optimization function is used to produce the appropriate k to capture the variety exists in the data points. At some value of k, when the function reaches a plateau, it is assumed that the starting point of the plateau is the optimal k [16]. However, in this framework, it is difficult to apply the stopping criterion [19], particularly, if no prior knowledge on dataset is available.

Progeny [20] is a clustering method that constructs new clusters out of existing ones based on their proposed stability metric. The measure of stability in this method is based on a co-occurrence probability matrix that verifies the appropriate cluster for new data points. The method starts with a range of possible solutions (*i.e.,* set of k values). The optimal value of k is determined based on the highest stability score. We compare ClustCrypt against with this method in Section VI.

Methods utilizing genetic algorithm [24] have shown to perform well in solving different optimization problems, such as clustering, by starting with a population of a set of potential solutions (chromosomes) and evolving towards a nearly optimal solution. Instead of providing several solutions to a particular problem, genetic algorithms keep a well balance between exploration (crossover) of the space of solutions and the exploitation (mutation) of the promising regions. However,
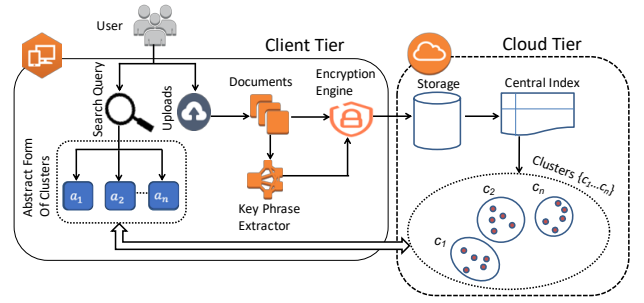

Fig. (1) Overall Search System Architecture Integrating Proposed Clustering Approach.

it is computationally prohibitive to utilize Genetic algorithms for clustering, as we need to consider trillions of combinations generated by tokens of a big dataset.

The idea of topic-based term clustering on an index structure (*i.e.,* data points) has been extensively studied on plain-text. Xu and Croft [36] proposed the idea that clusters formed for a homogeneous index (*i.e.,* an index that all of its terms share a similar topic) improve the efficacy of a search system in comparison to standard state-of-the-art information retrieval systems. The authors used k-means clustering method and distributed the indexed elements among the clusters using KL-divergence distance function [10]. Once the clusters are built, for an incoming search query, they perform pruning to determine the highest relevant cluster via utilizing the maximum likelihood estimation theory.

III. BACKGROUND

ClustCrypt is motivated by S3BD, a cloud-based secure semantic search system, that requires clustering over encrypted big data [21]. The search system can be used by law-enforcement agencies to detect criminal activities by searching over privacy-protected criminal reports.

Figure 1 presents where ClustCrypt is positioned within the S3BD system. We can see that S3BD is composed of a *client tier* and a *cloud tier*. The client tier is considered trusted and it provides *upload* and *search* functionalities for the users. The cloud tier is considered honest but curious, therefore, all the documents and their indexed tokens are stored in encrypted form. To enable real-time searching, the encrypted indexed tokens have to be clustered.

In the current version, S3BD repurposes k-means clustering to restrict the search space and provide real-time search operation over big data. However, the clustering is carried out based on a fixed k value, regardless of the dataset characteristics that limits its search quality. In this research, we propose an approach that considers the dataset characteristics and improves clustering of the encrypted keywords. Although we implemented ClustCrypt clustering approach in the context of S3BD, the approach is generic and can be deployed in other systems that require clustering of encrypted data.

As shown in Figure 1, once a user uploads documents, a keyword extractor is used to extract n keywords (aka tokens) from the original documents. S3BD fairly treats all documents and keeps the value of n constant across all documents in a dataset. Then, the documents and tokens are both encrypted and sent to the cloud tier. RSA deterministic encryption

technique [21] is used to encrypt documents and extracted tokens.

Cloud tier maintains a *central index* structure that includes key-value pairs. Each key-value pair represents, respectively, an encrypted token, and the list of documents where the token appears in, plus the frequency of the token in each one of those documents. Homomorphic encryption [18] can be used to encrypt the token frequency information. However, due to the slow down imposed by processing homomorphically encrypted data [18] and to practically maintain the real-time search quality, currently, the frequency information are stored in unencrypted form. Upon issuing a search query by a user, the search keywords are encrypted and searched against the central index in the cloud tier to retrieve relevant documents. Upon receiving the list of matching documents, the user can download and decrypt them utilizing his/her private key.

Clusters $c_1, ..., c_n$ are constructed based on the tokens of the index structure and to mitigate exhaustively searching the whole index structure for every single search query. The clusters are topic-based and they are constructed so that the union of the k clusters is equivalent to the index structure. For a given search query, instead of searching the whole index, the search space is pruned and gets limited to only those clusters that are topically related to the search query. The pruning is achieved based on a set of Abstract structures (denoted $a_1, ..., a_n$) that are sampled from each one of the clusters and reside either on the client tier or possibly on a trusted edge server [32]. Details of the way abstracts are created described in [21] [11]. Upon issuing a search query, the most similar abstracts to the search query are chosen and then, their corresponding clusters are searched.

IV. CLUSTCRYPT: CLUSTERING ENCRYPTED TOKENS

A. Overview

Topic-based clustering partitions indexed tokens based on their similarity. Due to encryption, however, the indexed tokens do not carry any semantic information that makes clustering a challenging task. To overcome this challenge, clustering is achieved based on *statistical semantics*. The idea is to locate tokens that are semantically close to each other in the same cluster. To achieve this, we first need to know the number of clusters (k) that should be created to cover topics exist in token of a given dataset. Then, we find the central tokens for each cluster and assign the rest of tokens to the most topically related clusters. In our work, according to S3BD system, we consider that the frequency and co-occurrences of all tokens in the dataset are available.

In the rest of this section, we first describe how to estimate the appropriate (k) for a given set of indexed tokens. Then, in Sections IV-C and IV-D, we explain the center selection and token distribution methods.

B. Estimating Number of Clusters (k)

Determining the appropriate number of clusters (k) is important, because it directly impacts the searching performance. Depending on the characteristics of a dataset and distribution of tokens in its documents, the value of k can vary significantly. Encrypted tokens, extracted from documents, and statistical data corresponding to them (*i.e.,* tokens' appearance in documents and their frequency) are available parameters to estimate k in the following manner:

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

TABLE (I)   Token-Document Matrix A obtained from index

Word	Hash	d_1	d_2	d_3	d_4	d_5	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 it to include only the token 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 (A), we only consider 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 i and document j, the maximum value in column j of the token-document frequency matrix represents the token with the highest association with j.

Therefore, for normalization, we divide the value of each entry of matrix A to the highest value in the corresponding column of the matrix and the result is stored in a matrix, called N. The value for each entry $n_{i,j}$ is formally calculated by $\frac{a_{i,j}}{\max\limits_{\forall i} a_{i,j}}$. For the example provided in Table I, the normalized matrix N is represented in Table II.

TABLE (II)   Normalized Token-Document matrix N

Word	Hash	d_1	d_2	d_3	d_4	d_5	d_6
Book	Uh5W	0.58	0	0.34	0.07	1	0
Solve	/Vdn	0.1	0	0	1	0.85	0
Traffic	oR1r	0	0.47	0	0.5	0	0
Net	vJHZ	1	1	0	0.38	0	1
Enter	tH7c	0	0.92	1	0	0.08	0.19

Step-3: Building Probabilistic Matrices R and 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. Our hypothesis is that tokens that co-occur across documents are likely to share the same topic. In addition, the magnitude of similarity between two given tokens could be influenced by distribution of the tokens 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 number of required clusters (k). We leverage the normalized matrix (N) to perform a two-phase probability calculation that ultimately yields a matrix (denoted as C) representing token-to-token topic similarity.

In the first phase, we calculate the *importance* of each token to each documents. The importance of token t_i, in document d_j denoted by τ_{ij} and is defined as $\tau_{ij} = n_{i,j}/\sum_{\forall k} n_{i,k}$. Considering each τ_{ij} and N, we generate R matrix whose entries represent the importance of each token across all documents. In fact, each entry $r_{i,j}$ of R matrix represents the probability of choosing a document d_j, having token t_i. That is, $r_{i,j} = P(t_i, d_j)$.

In our example, Table III shows the R matrix obtained from the N matrix (shown in Table II).

TABLE (III) R matrix: Built from N

Word	Hash	d_1	d_2	d_3	d_4	d_5	d_6
Book	Uh5W	0.29	0	0.17	0.04	0.50	0
Solve	/Vdn	.05	0	0	0.51	0.43	0
Traffic	oRir	0	0.48	0	0.52	0	0
Net	vJHZ	0.29	0.29	0	0.11	0	0.29
Enter	tH7c	0	0.42	0.45	0	0.03	0.09

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 δ_{ji} and is defined as $\delta_{ji} = n_{j,i}/\sum_{\forall q} n_{q,i}$. Considering each δ_{ji} and N, we generate S matrix 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 $s_{i,j} = P(d_j, t_i)$). In our example, Table IV shows the S matrix obtained from the N matrix (shown in Table II).

TABLE (IV) S matrix: Built from N

Docs	Book Uh5W	Solve /Vdn	Traffic oRir	Net vJHZ	Enter tH7c
d_1	0.34	0.06	0	0.60	0
d_2	0	0	0.19	0.49	0.38
d_3	0.17	0	0	0	0.45
d_4	.04	0.51	0.25	0.19	0
d_5	0.52	0.44	0	0	0.04
d_6	0	0	0	0.84	0.16

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

TABLE (V)   C matrix: Multiplication of R & S

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 C signify the topic similarity of each tokens with itself and separation from other topics. More specifically, the value of $c_{i,i}$ indicates the magnitude that term t_i does not share its topic with other terms. Therefore, we define diagonal entries ($c_{i,i}$) as *separation factor*, because for each token it represents the token's tendency to stay separate from other topics to the degree of the coefficient. 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 matrix C. Then, Equation 1 is used to approximate k for a dataset. We use the ceiling function to make k an integer value.

$$k = \lceil \sum_{i=1}^{m} c_{i,i} \rceil \qquad (1)$$

Correctness of k is verified using a hypothesis that states k within a set should be higher if individual elements of the set are dissimilar, otherwise k should be lower [14], [17]. Equation 1 is the core part of approximating k. According to this equation, the highest k could be m when each individual token represents a topic, otherwise it is lower. Hence, our approach conforms with the clustering hypothesis.

C. Determining Clusters' Centers

In k-means clustering, generally, the clusters' centers are arbitrarily chosen [9], [23]. Then, based on a distance measure function (*e.g.,* Euclidean distance [9] or semantic graph [23]), dataset elements are distributed into clusters. K-means operates based on iteratively shifting clusters' centers until convergence. 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 [8]. 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 be choosing the top k tokens from index that have the highest number of associated documents. Although this approach chooses important (highly associated) tokens, it selects centers with document and topical overlap. Alternatively, 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 descendingly 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* [21] 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 $c_{i,j}$ of matrix C represents the topic similarity between tokens i and j. Besides, diagonal entry $c_{i,i}$ measures separation of token i from others. Therefore, the total similarity token i shares with others can be obtained by $\Sigma_{\forall j | j \neq i} c_{i,j}$. Note that for token i, we have $\Sigma_{\forall j} c_{i,j} = 1$, hence, the total similarity for token i is equal to $1 - c_{i,i}$. 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 $\Phi_i = \omega_i \times c_{i,i} \times (1 - c_{i,i})$.

Algorithm 1: Determining clusters' centers

Input : k, C matrix, and *central index* (with tokens sorted descendingly based on the degree of document association)

Output: Set *centers* that includes at most k center tokens

1 **Function Choose Center($k,C,Index$) :**
2    $centers \leftarrow \emptyset$
3    $U \leftarrow \emptyset$
4    $\Theta \leftarrow \{(\emptyset, \emptyset)\}$ //Pairs of tokens and centrality values
5   **foreach** *token $i \in$ index* **do**
6       $\omega_i \leftarrow$ **CalculateUniqueness**(i,U)
7   **if** $\omega_i > 1$ **then**
8          $U \leftarrow U \cup A_i$
9          $\Phi_i \leftarrow (\omega_i \times c_{i,i} \times (1 - c_{i,i}))$
10   Add pair (i, Φ_i) to max-heap Θ based on Φ_i
11   **end**
12   **end**
13    $centers \leftarrow$ Extract k max pairs from Θ heap
14   **return** *centers*
15 **end**

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 C matrix as inputs. The algorithm returns a set of at most k center tokens, denoted *centers*, as output. We do not consider the lower bound in this case. The algorithm intuitively selects at most K- number of cluster centers. In the beginning, the output set is initialized to null. U represents the set of documents covered with the chosen centers. A max-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 to 12) and the corresponding pair is inserted to the heap. Note that tokens with uniqueness lower than or equal to 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.

D. Clustering Tokens

Once k tokens are chosen as cluster centers, the other tokens of the index are distributed among the clusters based on the relatedness (aka distance) between the centers and others.

Established techniques exist to calculate such relatedness (*e.g.,* semantic graph [23], Euclidean distance [9]), but they are not suitable for tokens that are sparsely distributed [9]. Besides, these are not designed to apply on encrypted data [23].

In S3BD [21], a method based on document co-occurrence is proposed to measure relatedness and cluster encrypted tokens. In this method, if two tokens are present in the same set of documents, they are considered related [21]. We utilize that to measure the relatedness of tokens with cluster centers and distribute tokens to the most related cluster. To determine the relatedness between a particular token and a center, we need to calculate the *contribution* and *co-occurrences* metrics for the token. Let t be a token in document d of dataset D with frequency denoted as $f(t,d)$. Then, contribution of d to t, denoted as $\kappa(d,t)$, is defined based on Equation 2.

$$\kappa(d,t) = \frac{f(t,d)}{\sum\limits_{j \in D} f(t,j)} \tag{2}$$

Co-occurrence of token t with center token γ_x in document d (denoted $\rho(t,d,\gamma_x)$) is defined as a ratio of the sum of frequencies of t and center γ_x in d to the total frequencies of t and γ_x throughout the dataset. The formal presentation of co-occurrence is provided in Equation 3.

$$\rho(t,d,\gamma_x) = \frac{f(t,d) + f(\gamma_x,d)}{\sum\limits_{j \in D} (f(t,j) + f(\gamma_x,j))} \tag{3}$$

Based on the contribution and co-occurrence metrics, relatedness between token t and γ_x (denoted $r(\gamma_x,t)$), is defined as multiplication of these two metrics (shown in Equation 4).

$$r(\gamma_x,t) = \sum_{j \in D} \kappa(j,t) \cdot \log\left(\rho(t,\gamma_x,j)\right) \tag{4}$$

In this equation, we iterate through each document j that is associated with t and consider the contribution of j to t and also the co-occurrences of token t and center γ_x through document j. We utilize the log function to constrain the impact of the co-occurrence factor.

V. Security Analysis

ClustCrypt is applicable in searchable encryption systems and, more generally, in encryption-based document retrieval systems. We envision that the systems utilize ClustCrypt are composed of at least two tiers (*e.g.,* client and cloud tiers in Figure 1). Only the actions performed on the client tier are considered trustworthy and clouds are considered honest but curious. We consider internal and external attackers desire to unveil the encrypted tokens and documents. In addition, attacks are possible in the communication between the client tier and cloud tier. To explain the threat model, we provide the following preliminaries:

View: The term 'view' denotes the portion that is visible to the cloud during any given interaction between client and server. The index and the set of clusters, the encrypted query (trapdoor) of a given search query (Q'), and the collection of encrypted documents D'. In some models, Q' also contains a particular weight for each term. The search result set for Q' is considered as I_c. Let $V(I_c)$ be this view.

Trace: It denotes the data exposed about I_c. Our aim is to minimize the data an attacker can infer from I_c.

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 [21]. The probabilistic encryption does not utilize one-to-one mapping and so, D' is not prone to dictionary-based attacks [34]. In addition, each token of the cluster is deterministically encrypted. Each cluster in the View only shows frequency of the encrypted tokens in documents in plain-text format. We note that even the frequency information can be stored in encrypted form using homomorphic encryption techniques [18], however, due to practical issues and maintain the real-time behavior of the system, we consider that as a future study.

If an attacker gains the access to the system, only he could understand the importance of a particular encrypted token by observing its frequency, but he/she cannot decrypt the token.

In an extreme scenario, let us assume that an attacker manages access to the key for deterministic encryption from the user's side. Theoretically, the attacker could build a dictionary considering all tokens from the clusters. Eventually, he/she tries to build a clone document set D'' utilizing the dictionary. Although all of the tokens extracted from a particular document are sufficient to understand the topic of the document, it is not possible to unveil the whole document.

VI. PERFORMANCE EVALUATION

A. Experimental Setup and Datasets

We implemented ClustCrypt within the S3BD context to cluster tokens in its central index. For evaluation, we compare and analyze the quality of clustering against other approaches that are both in encrypted and unencrypted domains. Our implementation of ClustCrypt is publicly available in https://git.io/fjf5X. For experiment, we used two 10-core 2.8 GHz E5 Xeon processors with 64 GB memory. To assure effectiveness of ClustCrypt, we evaluate it using three distinct datasets. These are selected based on their volume and characteristics of their data. To evaluate performance of ClustCrypt with big data, we use a subset of the `Amazon Common Crawl Corpus (ACCC)` dataset [4]. The whole dataset size ≈150 terabytes. The dataset is not domain-specific and contains different web contents, such as blogs and social media contents. Due to the limited processing capability of our available machine, we sampled from the dataset and randomly selected 6,119 documents that form a ≈500 GB document set. The second dataset, named `Request For Comments (RFC)`, is domain-specific and includes documents about Internet and communication networks. `RFC` includes 2,000 documents and the size is ≈ 247 MB. The third dataset is called `BBC` that is not domain-specific and includes popular news in various fields such as technology, politics, sports, and business. It contains 1000 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 [26] to identify keywords semantically represent the document.

To evaluate ClustCrypt, we instrument the pre-trained *Google News Word2vec* [27] model that determines the similarity between any two words. The model is a 300-dimension vector representation of three million words and phrases.
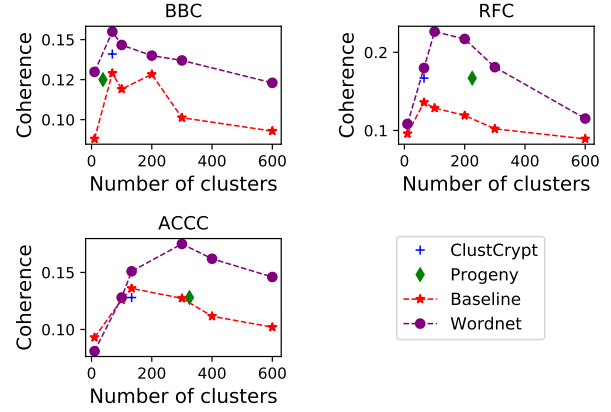

Fig. (2) Cluster coherency of ClustCrypt against encrypted (Progeny) and unencrypted (Baseline and Wordnet) approaches

Word2vec system requires a text dataset as input to build a vocabulary from the input dataset and learns vector representation of words. The model uses cosine similarity and provides the similarity score ($-1 \leq similarity\ score \leq 1$) for any two given tokens. We note that Word2vec model operates only on unencrypted tokens. Hence, to utilize Word2vec, we had to cluster the unencrypted tokens. That means, for evaluation purposes, we build the central index using plain-text tokens but ClustCrypt assumes tokens to be encrypted and does not use the plain-text form.

The performance metric that we use to evaluate the quality of topic-based clustering is overall *cluster coherency*. It represents how tokens in a cluster are related to a certain topic. For a given cluster, *coherence* is calculated based on mean of semantic similarity across all possible pairs of tokens in that cluster. Then, the mean of coherency across all clusters is defined as the overall cluster coherency.

B. Experimental Results

1) Clusters' Coherency: In this experiment, we determine the coherency of clusters resulted from ClustCrypt and compare it against those obtained from three other approaches. One is the conventional *k*-means (termed *baseline*). The second is an enhanced *k*-means, known as *Wordnet* [5], that generates a synonym set (termed Wordnet synsets) based on the input documents [28]. Then, *k*-means clustering is performed on the set of synsets. Token distribution in Wordnet is performed using edge counting method proposed by Wu & Palmer [28]. Both of these approaches operate on unencrypted data. The third approach is *Progeny* [20], which is a cluster estimation approach. To make Progeny comparable with others, we input its estimated *k* to Algorithm 1 to select clusters' centers.

Figure 2 shows results of the evaluation on the datasets. We note that both in baseline and Wordnet, *k* values are randomly chosen. We calculate and present clusters' coherency for all considered *k* values. Thus, for these two approaches, we have multiple data points in the figure. In contrast, ClustCrypt and Progeny are not iterative and have one data point. Using ClustCrypt, we obtain 133, 65, and 69 as *k* values for `BBC`, `RFC`, and `ACCC`, respectively. As `ACCC` is the largest and not domain specific, it yields the highest *k*. On the contrary, although `RFC`

is not the smallest dataset, because it is domain specific, it yields the lowest *k*. We observe that ClustCrypt results into the highest coherency for the `RFC` dataset. *Progeny* estimates 42, 225, and 340 clusters for `BBC`, `RFC`, and `ACCC`, respectively. However, after using Algorithm 1 for center selection and clustering, only 42, 65, and 69 clusters are built. According to the figure, Wordnet clusters have the highest coherency metric. In fact, it is difficult for an encrypted clustering approach (ClustCrypt) to outperform the unencrypted ones, because they do not have access to the semantic dictionary [28]. In contrast, in ClustCrypt, as the original meaning of data is unavailable, we populate clusters based on the co-occurrences. We observe that ClustCrypt competes with the baseline approach. In particular, in compare to the baseline approach, ClustCrypt provides a higher coherency for `RFC` and `BBC` datasets.

*2) **Analyzing the Impact of Dynamic Clustering**:* One goal of this research is to enhance performance of S3BD secure search system. As such, we implemented ClustCrypt within the context of S3BD and compared the coherency of resulting clusters with its original clustering approach, which is a pre-determined value for $k = 10$. Also, the center selection is only based on co-occurrences. In this experiment, we intend to evaluate the improvement that ClustCrypt achieves, when it is used within S3BD on the three aforementioned datasets. In this experiment, *k* estimated for `BBC`, `RFC` and `ACCC` are 69, 65, and 133, respectively.
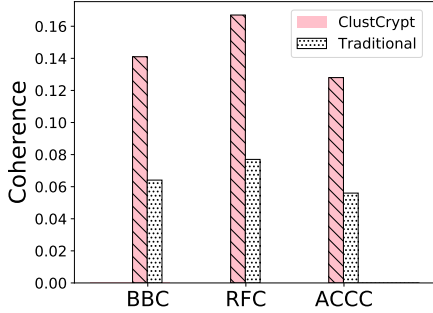
Fig. (3) Comparing the impact of dynamic clustering of ClustCrypt with static clustering in S3BD secure search system

Figure 3 shows that for all the studied datasets, clusters generated by ClustCrypt have remarkably higher coherency than the original static approach. This shows determining number of cluster dynamically based on dataset characteristics and choosing center tokens based on the centrality concept is effective. Such efficiency leads to more accurate and relevant semantic search results on big data, because tokens in clusters are more related to the topic of the cluster. For further analysis, next experiments concentrate on the impact of ClustCrypt on the quality of search results.

*3) **Analyzing the Impact of ClustCrypt on Search Systems**:* Quality of clustering tokens impacts both the accuracy and response time (*i.e.,* search time) of the search systems like S3BD. In fact, the aim of this study is to improve the clusters' coherency that impacts the search accuracy by retrieving more relevant documents. To evaluate the clustering impact, in this part, we compare and analyze how the search accuracy of S3BD system is affected by utilizing ClustCrypt's clusters
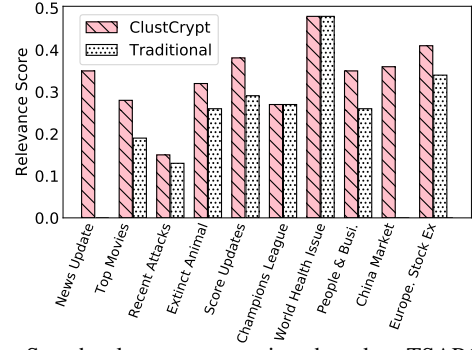
Fig. (4) Search relevancy comparison based on TSAP@10 scoring metric: ClustCrypt vs. original S3BD clustering

against the circumstance where the original static clustering is utilized. Then, we study the impact of clustering on the search time too. For evaluation, we generated a set of 10 benchmark search queries for each dataset, as listed in Table VI.

ACCC	BBC	RFC
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 (VI) Benchmark search queries for evaluated datasets

Impact on relevancy of the search results: To measure the relevancy of search results for each query, we use *TREC-Style Average Precision* scoring method [25]. This method works based on the recall-precision concept and 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 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 ClustCrypt against original S3BD clustering, we search the benchmark queries using S3BD. In Figure 4, the relevancy scores (vertical axis) of each query (horizontal axis) by utilizing the two approaches are represented. According to the Figure, for most of the queries, the TSAP@10 scores obtained by ClustCrypt clusters offer the higher relevancy. For two queries, the same *TSAP@10* score is offered, because the retrieved document lists are equivalent for them. Also, ClustCrypt clusters provide score for *news update* and *China Market* benchmark queries, whereas original S3BD clusters do not retrieve any relevant documents for them.

Impact on the search time: Figure 5 presents the overall search time of the benchmark queries for each dataset. This figure indicates that, in addition to providing more accurate results, ClustCrypt also offers a shorter search time. Longer search time impacts scalability and real-time quality of the search operation on big data. However, we can observe that in

compare to the clusters generated by original S3BD approach, ClustCrypt is more scalable in terms of real-time search operation.
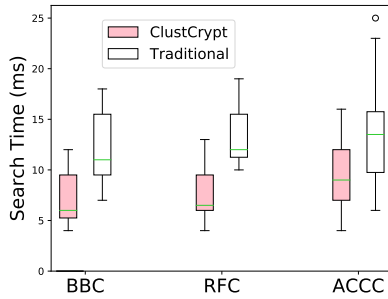

Fig. (5) Search time of ClustCrypt versus original S3BD clustering

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, we developed ClustCrypt, to efficiently perform topic-based clustering of unstructured encrypted big data in the cloud. ClustCrypt approximates the number of clusters for a given dataset with reduced time complexity, compared to conventional clustering approaches. Utilizing statistical meta-data, we obtain the probabilistic tendency of each token being segregated from others and use it to estimate the number of clusters. We leveraged the probabilistic analysis to determine center tokens and disseminate encrypted tokens to relevant clusters. Experimental results show that clustering using ClustCrypt provides on average 60% more coherency, comparing to conventional approaches used for encrypted data. We concluded that it is difficult for encrypted clustering to outperform plain-text clustering. However, ClustCrypt performs closer to the benchmark than other encryption-based clustering approaches. From the searchable encryption perspective, ClustCrypt helps to provide more relevant search results. In future, we plan to extend ClustCrypt to cluster growing datasets (*e.g.,* social networks). Also, we will investigate cluster pruning mechanisms to improve the search accuracy.

REFERENCES

[1] www.researchgate.net/post/What_is_the_time_complexity_of_clustering_algorithms. Accessed Aug. '18.
[2] www.money.cnn.com/2017/10/03/technology/business/yahoo-breach-3-billion-accounts/index.html. Accessed Feb '19.
[3] www.upguard.com/breaches/verizon-cloud-leak. Accessed Feb '19.
[4] http://commoncrawl.org. Accessed Feb'18.
[5] www.github.com/darenr/wordnet-clusters. Accessed May. 18.
[6] www.csoonline.com/article/2130877/data-breach/. Accessed Nov. '18.
[7] www.entrepreneur.com/article/273561. Accessed Nov. 18.
[8] 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)*, pages 420–434, Jan 2001.
[9] 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. 2019.
[10] Hirotogu Akaike. *Journal of Selected Papers of Hirotogu Akaike*, page 199, 2012.
[11] Sahan and Ahmad, SM Zobaed, Raju Gottumukkala, and MA 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*. IEEE, 2019.
[12] L. A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2):22–28, Mar. 2003.
[13] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. 2006.
[14] 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.
[15] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on parallel and distributed systems (TPDPS)*, 25(1):222–233, 2014.
[16] A. Coates and Andrew Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*, pages 561–580. 2012.
[17] Douglass R Cutting, David R Karger, Jan O Pedersen, and John W Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. 51(2):148–159, 2017.
[18] Léo Ducas and Daniele Micciancio. In *Proceedings of 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 617–640, 2015.
[19] P. Hammersley. Editorial information and information systems. *The Computer Journal*, 32(3):193, 1989.
[20] 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, 2015.
[21] M. A. Salehi J. Woodworth. S3BD: Secure semantic search over encrypted big data in the cloud. *Journal of Concurrency and Computation:Practice and Experience (CCPE)*, 28, 2018.
[22] Chih Lee, Ali Abdool, and Chun-Hsi Huang. Pca-based population structure inference with generic clustering algorithms. *Journal of BMC bioinformatics*, 10(1):S73, Jan. 2009.
[23] X. 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, 2004.
[24] R Lletı, M Cruz Ortiz, Luis A Sarabia, and M Sagrario Sánchez. Selecting variables for k-means cluster analysis by using a genetic algorithm that optimises the silhouettes. *Journal of Analytica Chimica Acta*, 515(1):87–100, 2004.
[25] 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, Dec. 2012.
[26] 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, Aug. 2009.
[27] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Conference on Neural Information Processing Systems, (NIPS)*, pages 3111–3119, Dec 2013.
[28] George A Miller. Wordnet: a lexical database for english. *Journal of Communications of the ACM*, 38(11):39–41, 1995.
[29] 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, 2015.
[30] 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, 2000.
[31] H. Pham, J. Woodworth, and M. A. Salehi. Survey on secure search over encrypted data on the cloud. *Accepted in Journal of Concurrency and Computation:Practice and Experience (CCPE)*.
[32] 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, 2017.
[33] Michael N Vrahatis, Basilis Boutsinas, Panagiotis Alevizos, and Georgios Pavlides. The new k-windows algorithm for improving thek-means clustering algorithm. *journal of complexity*, 18(1):375–391, 2002.
[34] Ding Wang and Ping Wang. Offline dictionary attack on password authentication schemes using smart cards. In *Information Security*, 2015.
[35] J. Woodworth, M. A. Salehi, and V. 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, 2016.
[36] 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, Aug. 1999.
[37] SM Zobaed, Md Enamul Haque, Shahidullah Kaiser, and Razin Farhan Hussain. Nocs2: Topic-based clustering of big data text corpus in the cloud. In *Proceedings of the 21st International Conference of Computer and Information Technology (ICCIT)*, pages 1–6, Dec. 2018.
[38] Sm Zobaed and Mohsen Amini Salehi. Big data in the cloud. In *Encyclopedia of Big Data . Edited by Albert Zomaya and Sherif Sakr, Springer International Publishing*.