Efficiency and Security in Similarity Cloud Services

Stepan Kozak
(supervised by Pavel Zezula)
Masaryk University, Brno, Czech Republic
xkozak1@fi.muni.cz

ABSTRACT

With growing popularity of cloud services, the trend in the industry is to outsource the data to a 3rd party system that provides searching in the data as a service. This approach naturally brings privacy concerns about the (potentially sensitive) data. Recently, quite extensive research of outsourcing classic exact-match or keyword search has been done. However, not much attention has been paid to the outsourcing of the similarity search, which becomes more and more important in information retrieval applications.

In this work, we propose to the research community a model of outsourcing similarity search to the cloud environment (so called *similarity cloud*). We establish privacy and efficiency requirements to be laid down for the similarity cloud with an emphasis on practical use of the system in real applications; this requirement list can be used as a general guideline for practical system analysis and we use it to analyze current existing approaches. We propose two new similarity indexes that ensure data privacy and thus are suitable for search systems outsourced in a cloud. The balance of the first proposed technique EM-Index is more on the efficiency side while the other (DSH Index) shifts this balance more to the privacy side.

1. MOTIVATION

With the rapid growth of the volume and diversity of digital data produced by all kinds of commercial, scientific and leisure-time applications, the retrieval in large data sets became one of the key IT tasks nowadays. The complex data types, such as multimedia or various sensor data, introduce a natural requirement to be searched not only by their metadata but also by the *content* of the data itself. This is typically beyond the capabilities of classic exact match or keyword search techniques and thus the use of various *similarity search* technologies increases significantly in current applications. A considerable research effort has been invested in this topic resulting in both theoretical background [24] and large-scale practical results [17, 3].

Proceedings of the VLDB Endowment, Vol. 6, No. 12

However, the similarity search is a very resource demanding process and the underlying technologies are rather complex. Therefore, there is a strong motivation to develop a general method that would provide the similarity search *as a service* to make the similarity search technology easily available to the users (data owners). As the popularity of cloud services is on the rise, the natural approach is to outsource this task to the cloud environment in a Software as a Service (SaaS) manner. We refer to this concept as *similarity cloud* [12]. This approach provides many advantages for the owners of the data, such as low initial investments, low storage costs and a very good scalability. Also, the cloud service should transfer the computational burden from the clients to the servers which would enable the clients to be simpler devices (such as cell phones).

On the other hand, the principle of cloud outsourcing fundamentally assumes that the data is provided to third party repositories that are shared between multiple tenants and which cannot be fully controlled by the data owner. The outsourced data may be sensitive (e.g. medicine data), confidential, or otherwise valuable (e.g. collected from a scientific research) and thus the privacy of the data is of high importance. Hence, besides providing effective and efficient searching, the similarity cloud also has to employ mechanisms to ensure data owner's privacy requirements not only by standard access permissions, but, more importantly, by securing the content of the indexed data in a potentially hostile third-party environment.

Ensuring privacy of the outsourced search is a widely studied topic in the context of classic databases. There has been a lot of focus on *symmetric searchable encryption* schemes that can form basic building blocks of secure cloud storages. Such schemes allow data owners to encrypt the data in such a way that it is possible to perform selective data retrieval (search) over the encrypted collection while the data privacy is ensured. Recently, Kamara et al. [11] described the requirements of a practical secure cloud storage considering the general case of relational databases and they proposed a symmetric searchable encryption for this scenario [10].

These general principles can also be applied in the context of similarity search, however, this area has several specifics which make the outsourcing more complex. First of all, the similarity search often deals with more than "one level" of the data to be processed: The *raw data* (for example a set of images) is typically preprocessed to obtain so called *features* (descriptors) that are indexed and searched (for example SIFT features from images [14]). From the privacy point of view, it is crucial to ensure privacy of both the raw data

and the indexed data objects (features), which may be, in some cases, equal to the raw data objects or can be highly correlated with them.

The second difference from the standard search techniques lies in the core of the similarity search – there exist an infinite number of *(dis)similarity functions* that can be used with a wide variety of data types. When searching the data, the similarity query typically contains an *example object* (query object) and the search should return the data objects that are the "most similar" to the example according to the specified similarity function. Since our goal is to develop a solution for a wide class of the similarity functions (specifically, metric functions), we cannot assume practically anything about the specific similarity. To a large extent, we have to consider the data and the function as *black box* and if the precise similarity of two objects should be measured, the only way is to provide the function with the original unencrypted data objects. This is a difference from the standard solutions where the knowledge of the data structure and the search model can be exploited better.

There has been some recent research focusing on the problem of outsourcing similarity search [13, 23, 12], however, none of these works consider the problem in the wide context of complex requirements of a practical similarity cloud system. Specifically, the authors focus mainly on the security aspects of the system and do not thoroughly evaluate the efficiency of their schema (client side, server side and their communication). Also, the authors do not consider the typical case in real applications where the data is dynamic and need to be updated often after the encrypted index is built. The aim of this work is to fill these gaps and to provide general definitions and requirements of the practical similarity cloud system as well as to propose techniques suitable for use in real applications.

First, we describe and "delimit" the wide problem of providing the (metric-based) similarity search as a service via outsourcing to the cloud environment (Section 2). Namely, we define a general architecture of the *similarity cloud* and its efficiency and security requirements. In Section 3, we describe existing related approaches and we discuss their levels of privacy and efficiency in the wider context of our general scheme. Further in Section 4, we propose our own solutions. The paper is concluded in Section 5.

2. CONCEPT OF SIMILARITY CLOUD

Let us introduce and describe the concept of similarity cloud as we see it and propose it to the research community. First, we mention fundamental terms used in the field of similarity search, then we describe the general model of the similarity cloud, and finally we formulate requirements that should be met by a successful similarity cloud.

The concept of similarity search is applicable to a wide range of data types together with a practically infinite number of various similarity functions. We adopt the general model of data that is suitable for many applications. The model is based on mathematical abstraction of metric space. The *metric space* is an ordered pair $\mathcal{M} = (M, d)$, where M is a *domain* of data objects and d is a total *distance function* $d : M \times M \longrightarrow \mathbb{R}$ satisfying metric postulates of non-negativity, identity, symmetry, and triangle inequality [24]. The set of indexed objects $X \subseteq M$ is typically searched by the *query-by-example* paradigm, for instance by the *range query* $Range(q, r) = \{o \in X \mid d(q, o) \leq r\}$, $q \in M$ or by the *nearest neighbors query* k-NN(q) covering k objects from X with the smallest distances to given $q \in M$. The following terminology is used throughout this paper:

Raw data is the original data to be indexed and searched, e.g., binary image files or other multimedia data. The domain of this data will be denoted as R.

Metric space objects are features extracted from the raw data; each MS object $o \in M$ is paired with the corresponding raw object $o_{raw} \in R$ by a unique ID; they are compared by a metric function d.[1]

Data owner is a subject outsourcing the search service; *authorized client* is any entity authorized by the data owner to use the search service (entity having the secret key).

By term *server(s)* we refer to the server(s) of the 3^{rd} party similarity cloud providing the services; from the data owner's point of view, the servers are not trusted because they are not fully under the data owner's control.

Trapdoor is an operation token generated by a "one-way" trapdoor function; in our setting, the trapdoor is used for communication between individual parties (in order to process a specific operation – search, insert, etc.) and thus it can be captured by an attacker; therefore, this token has to leak as little information as possible, but still contain enough information for the server to process the task.

2.1 General model

In the classic search paradigm, the search is realized directly in the actual raw data, whereas in the case of metric-based similarity search, the raw data often has to be preprocessed to extract the metric space objects which are the objects being indexed and searched. This difference between the classic and the similarity search approaches is even more significant from the privacy point of view, because the raw data and MS objects are usually highly correlated. Hence we need to ensure privacy not only for the raw data itself, but also for the MS objects.

This two-level nature of the similarity search inherently introduces several relatively independent problems. First, how to extract the MS objects on the server side while preserving the privacy of the raw data, second, how to provide efficient, privacy-preserving similarity search over the extracted MS objects, and third, how to securely store the raw data objects and provide them to the authorized clients. The similarity cloud is composed of three theoretically independent servers: secure storage of the (encrypted) raw data, server that extracts MS objects from the raw data, and the indexing server that organizes and searches the MS objects in a similarity manner. The general scheme of the similarity cloud is schematically depicted in Figure 1. Our model builds on the Similarity Searchable Symmetric Encryption which has been published by Kuzu et al. in the context of their Secure LSH Index [13] and generalized in the work of Tang [21]. However, their definition does not reflect an important requirement of "dynamicity" of the scheme (i.e. the technique should support efficient insertions and deletions of items after the index is built). We consider this update operations crucial for any practical scheme. Also, the model by Kuzu et al. does not consider the two-level character of the data searched by similarity (raw data vs. MS objects).

[1] In some cases, the raw data and the MS objects are identical (for instance, gene sequences or other biomedical data); in other cases, several MS objects are extracted from one raw object.

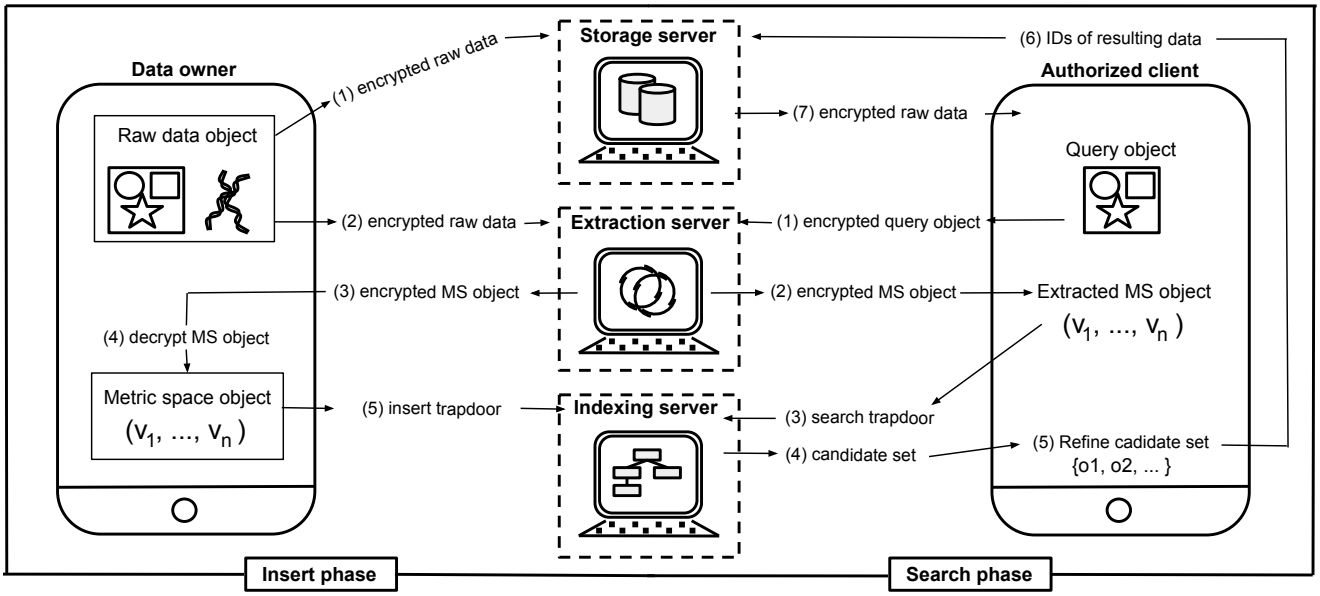

Figure 1: Similarity Cloud Scheme

The search query processing within the similarity cloud can be described as follows: Let $q_{raw} \in R$ be a raw data query object which is to be searched in the similarity cloud. First, the MS query object q is extracted from q_{raw} using the extraction server, this q is then used to obtain the search trapdoor on the client side. The trapdoor information is then sent to the indexing server which retrieves the candidate set of encrypted MS objects and sends them back to the client. Finally, this candidate set is decrypted and refined by the client and the result object IDs are passed to the storage server to obtain corresponding encrypted raw data which are decrypted to obtain the final query result.

Updates to the system are realized as follows: Let o_{raw} be the raw data object to be inserted into the system. First, the MS object o is extracted from o_{raw} (using the extraction server) and the original object o_{raw} is encrypted and sent to the storage server. Then, the insert trapdoor is generated from o and sent to the indexing server to update the index structure. Algorithm for data removal would be analogous.

Please, note that this scheme is general and in some scenarios not all the entities are employed. If the raw data space is directly the search metric space, e.g. strings with edit distance, then the extraction phase is skipped. In some cases, the raw data may not be stored on a separate server, but as a part of the secure index on the indexing server.[2]

2.2 Similarity Cloud Assessment Criteria

We expect the problem of secure similarity search to become more and more relevant and more authors proposing new techniques. To compare the techniques and to evaluate whether the specific technique is suitable for specific needs of particular data owner, we introduce a set of criteria as a guide according to which the similarity cloud solutions could be evaluated.

Usability criteria. From the usability point of view, it is important to answer the following questions about a specific similarity cloud technology: Can the solution be used for generic metric space or for a smaller class of data types? What kind of queries does the scheme support (range, k-NN, approximate k-NN)? Is the scheme dynamic (efficient insert, delete operation)?

Once the data owner found a solution satisfying the usability requirements, the main evaluation criteria of the scheme would be the level of privacy and system efficiency.

Security criteria. From the privacy point of view, servers of the cloud provider should learn as less information about the outsourced data as possible. The overall privacy of the system can be analyzed per partes in the following way:

Extraction privacy considers the information about raw data which leaks during the extraction on the remote server.

Privacy of the indexing process evaluates the amount of information the server can learn about the MS objects from the insert trapdoors received during index build or updates.

Search privacy takes into account possible information leakage from the search trapdoors (the server should not be able to get any information about the query parameters, especially the query object itself; moreover, only authorized clients can issue a meaningful query, i.e. the search trapdoor can be generated only by authorized clients).

Finally, *query result privacy* measures correlation between the search trapdoor and the resulting candidate set should not reveal any useful information to the server.

The resistance of the scheme can be evaluated using standard attack scenarios: Ciphertext-only Attack, Known Plaintext Attack or (Adaptive) Chosen Plaintext Attack [15].

In the real-world scenarios, it is also desirable if the scheme

[2]This is always possible, however merging these two functions has usually the disadvantage of revealing the correspondence between the search trapdoors and the encrypted result for such trapdoor. This makes the scheme more vulnerable to the statistical attacks.

provides guarantees on the data integrity (i.e. the scheme provides protocols by which the data owner can detect any unauthorized modification of the data or modification of the query answer by any potential adversary). The efficient verification protocols has been addressed in work of Goodrich et al. [6]. Also, the key management options of the scheme may become important, specifically, an option of (efficient) revoking of the secret keys for clients the data owner do not want to be authorized for the search anymore (without the need of re-encrypting and re-indexing the whole data).

Efficiency criteria. When evaluating efficiency of search operations, we should always consider the client-server nature of the system. In real scenarios, the time spent on the client is of a higher importance than the time of the query processing on the server(s). As the clients can be simple mobile devices with limited computational resources, it is crucial to move as much work as possible to the server side. Also the communication costs between client and server play an important role. These efficiency criteria should be also considered for the index build and update operations. Especially, an important aspect is, whether the scheme allows to add (or remove) an object without the need of re-indexing the whole data set.

Intuitively, the security requirements go against the efficiency objectives. If most of the computations should be performed on server side, the server has to have enough information about the data to process such task efficiently. Hence, the right balance between the security and efficiency should be found for each specific application setting.

3. EXISTING SOLUTIONS

The problem of outsourcing similarity search has been recently studied in several publications. The authors have focused on two separate problems which need to be solved on the way towards the secure similarity cloud. First, secure outsourcing of the MS objects extraction and second, building efficient secure structures for metric space indexing.

Pioneering work on secure extraction of image features in the 3rd party cloud environment has been published in a series of consecutive papers by Hsu et al. [7, 8, 9]. Using a homomorphic Paillier encryption scheme [20], the authors constructed so called privacy-preserving SIFT (Scale-invariant Feature Transform) which gives comparable results as the original SIFT but also provides the data owner with privacy guarantees and it is therefore suitable to be employed within the similarity cloud. For more details we refer the reader to the original publications [7, 8, 9].

Further in this work, we focus on the second issue – development of a secure similarity index. This has been recently addressed by several authors [13, 23, 12, 22]. In this section, we overview the proposed techniques in detail.

3.1 Encrypted Hierarchical Index

Encrypted hierarchical index (EHI) [23] is a relatively general concept where the server simply serves as the storage of encrypted nodes of any indexing structure. Client in turn requests encrypted nodes from the server which are necessary for search operation, decrypts them and proceeds further until the sufficient result is found.

According to the authors, the EHI method reaches perfect security, because a potential attacker cannot gain any information either about the data itself or about the search space, since everything is stored encrypted, in a flat structure of encrypted nodes. However, assuming that the indexing and searching algorithms are public, an attacker (i.e. the server) can learn the hierarchy of the underlying structure after several search operations, because the client typically iteratively requests all descendants of the currently processed node.

Naturally, the high security level of EHI comes at the cost of high communication (a lot of traffic is between client and the server) and of a relatively low search efficiency. Since all nodes of the indexing structure are encrypted, all time-consuming search operations have to be realized on the client side, and also the client has to perform a lot of encryption/decryption operations. Moreover, this scheme does not support efficient index updates. To insert/remove items to/from the index, the client has to either keep or request a significant part of the index so it can be updated, re-encrypted and uploaded to the server. For instance, if the root of the index is to be split, this operation could be extremely costly.

3.2 Metric Preserving Transformation

Another solution proposed in the work of Yiu et al. [23] is called Metric-Preserving Transformation (MPT) which uses an order-preserving encryption [1]. The main advantage of this approach is that although original distances are not kept on the server side in plain form, the distance comparisons can be correctly evaluated at the server side and therefore a simple filtering during the search phase is possible. However, to reach sufficient security level (of the encrypted distances), it is necessary to have a significant sample of the indexed data before the indexing structure is built. This could be a problem for dynamically changing data sets.

The secret key of the proposed technique contains a set of *anchor objects* (pivots) and each data object is assigned to its closest pivot [23]. This indexing structure actually reveals information about the search space by not hiding the number of items belonging to individual anchors (buckets). Since the radius of each bucket is encrypted using an order-preserving function, it is possible to compare bucket sizes (radii), observe the ratio between these sizes and radii, and thus identify dense areas in the data space. In the context of adaptive chosen plaintext attack, we can actually see a way to compromise the secret key of the scheme (i.e. reveal the anchor objects). The attacker can exploit the order-preserving function properties in the following way: For each pair of known objects o_1, o_2 belonging to the same anchor a_1, we know their encrypted distance to the anchor and we can determine which object is closer to the anchor just by comparing the encrypted distances. Therefore, we can determine to which side of the hyperplane defined by o_2 and o_1 anchor a_1 belongs. With more and more known objects, the attacker is able to locate a_1 more and more precisely.

3.3 Secure LSH Index

The latest published secure index suitable for metric-based similarity search is the Secure LSH Index (SLSH) [13] based on the concept of locality sensitive hashing (LSH) [5].

Main advantage of the SLSH index is its provable adaptive semantic security where the amount of information leaking to a 3rd party is clearly defined and it is proved to be the maximum amount that can leak. A disadvantage is the necessity of having a specific family of LSH functions for given

search space. Unfortunately, there are many spaces used in applications for which no LSH family is known, for instance metrics for evaluation of the multimedia data similarity such as Quadratic Form Distance (QFD) [24].

In such metric spaces it is not possible to directly apply SLSH indexing technique and there has to be an extra initial phase which transforms the objects from the original metric space to a different space with a known LSH family. This phase slows down the whole process (especially the index building phase) and, more importantly, the embedding into a different metric space inherently introduces an imprecision in the similarity search process. Further, the authors of Secure LSH index do not explicitly consider the scheme in the dynamic environment where objects are frequently added to or removed from the indexed set.

4. PROPOSED SOLUTIONS

The above described techniques differ not only in the level of privacy and efficiency, but also in the operations natively supported. The supported type of queries of individual approaches (including the EM-Index and Dynamic Secure Hash-based Index (DSHI) introduced later in this paper) are summarized in Table 1.

Table 1: Operations supported by the approaches

	k-NN	Approx. k-NN	Range query	Update
EHI [23]	+	+	+	-
MPT [23]	+	-	-	partially
FDH [23]	-	+	-	+
SLSH [13]	-	+	-	partially
EM-Index	+	+	+	+
DSHI	-	+	-	+

In our work, we aim at creating a general secure metric-based similarity index that would meet most of the criteria defined in the Section 2.2. Currently we work on two innovative approaches, the Encrypted M-Index (EM-Index) and the Dynamic Secure Hash-based Index (DSHI). Within the context of Similarity Cloud as defined in Section 2, our approaches implement all methods of client and indexing server and they seek a balance between the efficiency and privacy requirements.[3] The important feature of our techniques is the support of efficient dynamic operations.

4.1 Encrypted M-Index

Encrypted M-Index (EM-Index) is a general technique that adds the data privacy level to any metric indexes that are based on permutations of a fixed set of reference objects (pivots) [2, 18]. Specifically, an object $o \in X$ is indexed by identifiers of several closest pivots from o ordered by the distance from o (we will denote this as *pivot permutation*). We introduce our approach as an extension of one of these structures called M-Index [16, 18], which partitions the search space using dynamic Voronoi partitioning and enables both precise and approximate similarity search.

[3]Please note that neither EM-Index nor DSHI provide a mechanism for outsourcing raw data features extraction; we assume this extraction service to be available as a part of the client or outsourced to a separate extraction server.

Our solution exploits the fact that the prefix of pivot permutation is the only information necessary for navigation within the M-Index Voronoi cell tree [16]. Especially, no other distances are computed during the insertion phase and the algorithms for retrieving candidate set S_C for all considered types of similarity queries also need only the permutation prefixes or query-pivot distances. Therefore, the general idea of EM-Index is to make the distance function and the pivot set part of the private information known by the data owner (and shared with authorized clients); the second part of this secret key is a symmetric cipher key to encrypt the MS object data. The indexing server manages a dynamic Voronoi cell tree and stores the encrypted MS objects.

The prefix of pivot permutation is the main part of the trapdoor information for all insert, remove and search operations (range, k-NN and approximate k-NN queries). Additionally, the trapdoor for precise k-NN and range queries contains also the full object-pivot distances. Naturally, for the insert operation, the encrypted MS object data is also part of the trapdoor (an arbitrary symmetric cipher such AES, for instance, can be used).

One of advantages of the EM-Index (comparing to other approaches) is that all the indexing server procedures are standard, unmodified operations of the M-Index (or any other pivot-permutation technique). Thus, our approach can be used to introduce privacy guarantees to an existing index without any implementation changes of the working methods. We refer the reader to the already published paper [12] for more detailed description of the EM-Index, analysis of its security, and experimental evaluation of its efficiency using a prototype implementation [4].

4.2 Dynamic Secure Hash-based Index

The EM-Index is a general, scalable and efficient solution, however it may not be sufficient from the privacy point of view in some scenarios [12]. Therefore we propose the Dynamic Secure Hash-based Index (DSHI), where the balance between privacy and efficiency is more on the privacy side.

Currently, we are working on the DSHI proposal with the aid of the SLSH index introduced in Section 3.3 (in the two-server variant with a Paillier cryptosystem) [13]. As mentioned above, there are two main issues about the original SLSH index to be tackled. The first problem is that for metric spaces without a family of LSH functions, there has to be an initial phase of metric space embedding. This introduces further search imprecision and it negatively affects efficiency of the index building phase (and also search phase). To address this issue, we could again take the advantage of the indexing technique M-Index [18] which internally defines a generic hashing function satisfying the general LSH properties [19]. We plan to use several orthogonal M-Indexes as the LSH functions which would enable to use the rest of the technique for all the metric spaces.

The second problem of the original SLSH is inefficiency of update operations. We will propose a modification of the index that, exploiting the two-server variant of SLSH approach [13] together with Paillier cryptosystem [20], enables to to implement efficient dynamic operations (insertions, deletions) on the top of the M-Index-based SLSH. Both proposed extensions of the original SLSH preserve its privacy guarantees but add the necessary functionality to be better applicable in practical similarity cloud solutions.

5. CONCLUSIONS AND FUTURE WORK

We proposed a general concept of providing similarity search as a service. This similarity cloud definition focuses on preserving the efficiency of the search while ensuring privacy of the data passed from the data owner to the 3rd party cloud provider. Following the concept of similarity searchable encryption, we formally defined its security requirements. We provided extensive summary of currently existing solutions and evaluated them against the practical requirements of the similarity cloud.

Further, we briefly introduced a new method that can be used to ensure data privacy in similarity search systems outsourced in a cloud. This technique EM-Index exploits existing efficient metric indexes based on a fixed set of pivots. Except for precise and approximate k$-NN$ queries, it also supports precise evaluation of the range queries and efficient update operations. Currently, we work on a new Dynamic Secure Hash-based Index which would guarantee higher privacy level while still being suitable for dynamically changing data (i.e. technique supporting efficient index updates which we consider a must-have for a practical similarity cloud).

The ultimate goal of our research is to provide a complex, scalable similarity cloud solutions with provable security where the balance between efficiency and level of privacy can be shifted according to requirements of the specific domain and application.

6. REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order Preserving Encryption for Numeric Data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574, 2004.

[2] G. Amato and P. Savino. Approximate similarity search in metric spaces using inverted files. In *Proceedings of the 3rd international conference on Scalable information systems*, page 28, 2008.

[3] M. Batko, F. Falchi, C. Lucchese, D. Novak, R. Perego, F. Rabitti, J. Sedmidubsky, and P. Zezula. Building a Web-scale Image Similarity Search System. *Multimedia Tools and Applications*, 47(3):599–629, 2010.

[4] M. Batko, D. Novak, and P. Zezula. MESSIF: Metric Similarity Search Implementation Framework. In *Digital Libraries Research and Development*, volume 4877 of *LNCS*, pages 1–10. Springer, 2007.

[5] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, 1999.

[6] M. T. Goodrich, R. Tamassia, and N. Triandopoulos. Super-efficient verification of dynamic outsourced databases. In *Proceedings of the 2008 The Cryptopgraphers' Track at the RSA conference on Topics in cryptology*, pages 407–424, 2008.

[7] C. Hsu, C. Lu, and S. Pei. Homomorphic Encryption-based Secure SIFT for Privacy-Preserving Feature Extraction. In *Proceedings of SPIE*, volume 7880, page 12, 2010.

[8] C. Hsu, C. Lu, and S. Pei. Secure and robust SIFT with resistance to chosen-plaintext attack. In *Proceedings of 2010 IEEE International Conference on Image Processing*, pages 997–1000, 2010.

[9] C. Hsu, C. Lu, and S. Pei. Image Feature Extraction in Encrypted Domain with Privacy-Preserving SIFT. *IEEE Transactions on Image Processing*, pages 4593–4607, 2012.

[10] S. Kamara, P. Charalampos, and R. Tom. Dynamic Searchable Symmetric Encryption. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 965–976, 2012.

[11] S. Kamara and L. Kristin. Cryptographic Cloud Storage. In *Proceedings of the 14th international conference on Financial cryptograpy and data security*, volume 6054 of *LNCS*, pages 136–149, 2010.

[12] S. Kozak, D. Novak, and P. Zezula. Secure Metric-Based Index for Similarity Cloud. In *Proceedings of the 9th VLDB Workshop on Secure Data Management 2012*, volume 7482 of *LNCS*, pages 130–147, 2012.

[13] M. Kuzu, M. S. Islam, and M. Kantarcioglu. Efficient Similarity Search over Encrypted Data. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 1156–1167, 2012.

[14] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157 vol.2, 1999.

[15] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*, volume 106. 1997.

[16] D. Novak and M. Batko. Metric Index: An Efficient and Scalable Solution for Similarity Search. In *Second International Workshop on Similarity Search and Applications (SISAP 2009)*, pages 65–73, 2009.

[17] D. Novak, M. Batko, and P. Zezula. Generic similarity search engine demonstrated by an image retrieval application. In *Proceedings of ACM SIGIR'09*, page 840, New York, New York, USA, 2009.

[18] D. Novak, M. Batko, and P. Zezula. Metric Index: An Efficient and Scalable Solution for Precise and Approximate Similarity Search. *Information Systems*, 36(4):721–733, 2011.

[19] D. Novak, M. Kyselak, and P. Zezula. On locality-sensitive indexing in generic metric spaces. In *Proceedings of the Third International Conference on Similarity Search and Applications*, pages 59–66, 2010.

[20] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Proceedings of the 17th international conference on Theory and application of cryptographic techniques*, pages 223–238, 1999.

[21] Q. Tang. Search in Encrypted Data: Theoretical Models and Practical Applications. *IACR Cryptology ePrint Archive*, page 648, 2012.

[22] B. Yao, F. Li, and X. Xiao. Secure Nearest Neighbor Revisited. In *Proceedings of 29th IEEE International Conference on Data Engineering (to appear)*, 2013.

[23] M. L. Yiu, I. Assent, C. S. Jensen, and P. Kalnis. Outsourced Similarity Search on Metric Data Assets. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):338–352, 2012.

[24] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach*, volume 32. 2006.