Distance-Based Data Mining Over Encrypted Data

Christine Tex, Martin Schäler, Klemens Böhm

Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

firstname.lastname@kit.edu

***Abstract*—When mining data, organizations rely on service providers to carry out the analyses. However, data owners often are only willing to transfer their data when it is encrypted. So encryption must preserve the mining results. Since many mining algorithms are distance-based, we propose the notion of distance-preserving encryption (DPE). Designing a DPE-scheme is challenging, as it depends both on the data and the distance measure in use. We propose a procedure to engineer DPE-schemes, dubbed KIT-DPE. In a case study, we instantiate KIT-DPE for SQL query logs. We design DPE-schemes for all SQL query-distance measures from the literature. For all these measures, we prove that one can use a combination of existing property-preserving encryption schemes with known security characteristics to guarantee the same mining result.**

I. Introduction

Virtually any organization is reluctant to pass on its data to other parties. For instance, think of data analysis as a service, possibly in the cloud. An organization would be much more relaxed if, instead of uploading the original confidential data, it could pass on the encrypted counterpart. Depending on the application, encryption must preserve application-specific properties of the data [1], like for instance the order [2].

An important property of a data set is the pairwise distances between data items, since many data-analysis algorithms only rely on these distances. Examples are distance-based clustering [3], [4], [5] or outlier detection [6]. If encryption preserves these pairwise distances, mining results on the encrypted and on the plain-text data are the same. But distance-preserving encryption has not been investigated systematically before.

This paper examines how to design distance-preserving encryption (DPE) schemes for data of arbitrary type. For data consisting of items with a complex inner structure, such as graphs or query logs, this is challenging, for two reasons: (1) unclear subject of encryption and (2) distance-measure variety. We demonstrate this using SQL query logs. Such logs contain valuable information on user interests and are a useful resource for data mining. The mining of SQL query logs is involved [7], and it is reasonable to outsource it to a service provider.

Challenges: When designing DPE-schemes, one faces two challenges. First, for data with a complex structure like SQL queries, it may be unclear what "encryption of data items" actually means (cf. Example 1). Once this is solved, one must specify a security model tailored to the type of data considered and an encryption scheme in line with this model.

Example 1 (Unclear Subject of Encryption)**.** *Think of an SQL query log. Which parts of the query should be encrypted? Viable options are that one encrypts the query string as a whole – or only specific tokens such as values.*

Second, for any type of data, there exist different distance measures ("distance-measure variety"). Example 2 illustrates that distance-preserving encryption depends on the measure in use. While the second measure in the example requires an executable query, the first one does not. Thus, when designing a DPE-scheme, one must differentiate between the measures.

Example 2 (Distance-Measure Variety)**.** *For distance-based data mining over an SQL query log, different distance measures exist, with different intuitions of distance. For instance, one can use a string-distance measure like the Levenshtein distance or a measure that quantifies the overlap of tuples in the query results.*

Contributions: In this paper, we examine how to design distance-preserving encryption schemes for data with a complex structure. Our paper consists of two parts: (1) We introduce the general concepts, and (2) we study how to instantiate the concepts using the example of SQL query logs. First, in Part (1), we define distance-preserving encryption for arbitrary data and distance measures. Next, we propose a general procedure for designing distance-preserving encryption, named KIT-DPE. In Part (2), we study how to encrypt SQL queries preserving distances using our KIT-DPE procedure. As a result, we find DPE-schemes for four well-known SQL query-distance measures. They have a higher security level than schemes that the related approach CryptDB [8] would generate, i.e., shield against more attacks.

II. Background and Related Work

In this section, we explain how security of encryption schemes is defined, and which classes of property-preserving encryption schemes exist. We apply combinations of these schemes to realize distance-preserving encryption.

1) Security of Encryption Schemes: The security of an encryption scheme is measured by the success of an attacker with a specific attack. One differentiates between active and passive attacks. In our scenario, only passive attacks, e.g., eavesdropping, are relevant. Passive attacks are divided in cipher-text only, known-plain-text, and chosen-plain-text attacks [9]. For instance, in a cipher-text only attack, the attacker has access to several cipher-texts someone else has selected and tries to decrypt (any) new randomly selected cipher-text.

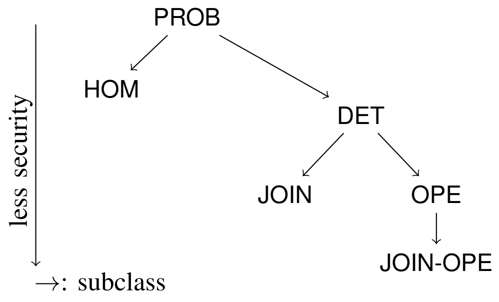

Fig. 1. Taxonomy of Property-Preserving Encryption Classes.

2) Property-Preserving Encryption: Property-preserving encryption schemes are classified by the plain-text properties they preserve. Fig. 1 depicts a taxonomy showing the relationships and the security levels of various classes of property-preserving encryption classes, inspired by [8], [10]. The rows stand for the security levels, where higher is better. For classes in the same row, a security ranking is not possible. We now explain the definitions of the classes.

Probabilistic Encryption (PROB). Encryption schemes are *probabilistic* if, in general, two equal values are mapped to different cipher-texts.

Homomorphic Encryption (HOM) [11]. Homomorphic encryption is a subclass of probabilistic encryption and allows computations of arithmetic aggregate functions such as the sum over encrypted data.

Deterministic Encryption (DET). Encryption schemes are *deterministic* if two equal values are mapped to the same cipher-text.

Order-Preserving Encryption (OPE) [2]. Order-preserving encryption schemes are deterministic and preserve the order of the data items.

Join-Preserving Encryption (JOIN/JOIN-OPE) [8]. JOIN is a special usage mode of a DET or OPE scheme, allowing to compute joins over encrypted data.

For every class, different instances (i.e., encryption schemes) exist. For instance, the randomized AES encryption scheme [12] is an instance of PROB. However, so far it is not known how one can ensure distance-preserving encryption for complex data using these classes. First, this is challenging as one usually has to consider different distance measures. Second, using known classes and schemes has the advantage that their security is well-studied, and thus, no further security analysis is necessary. Thus, we focus on how to realize distance-preserving encryption using these classes.

III. Distance-Preserving Encryption

Now we define distance-preserving encryption and Kit-DPE, our procedure for designing such encryption schemes.

A. Definition of Distance-Preserving Encryption

With distance-preserving encryption (DPE), the pairwise distances for the plain-text and the cipher-text data items must be the same, see Definition 1.

Definition 1 (Distance-Preserving Encryption (DPE))**.** *Let $\mathcal{D}$ be a data set and* $\mathtt{Enc}$ *an encryption algorithm for data items in $\mathcal{D}$. Then,* $\mathtt{Enc}$ *is d-distance preserving if*

$$\forall x, y : d(\mathtt{Enc}(x), \mathtt{Enc}(y)) = d(x, y).$$

Distance-preserving encryption enables distance-based data mining on encrypted data. This means that the mining results on cipher-text and on plain-text data are the same. For instance, data items are assigned to the same clusters.

B. Kit-DPE-*Procedure for Designing* DPE-*Schemes*

Now we introduce the general procedure for designing distance-preserving encryption (Kit-DPE) for complex data. Its four steps are as follows:

1) Definition of the Security Model: First, one has to specify the security goals, like "the SQL log should not reveal any information on the content of the database". To this end, one has (1) to specify the threat model, i.e., the attacks to shield against, and (2) to define a high-level encryption scheme for the type of data considered, e.g., "encrypt all constants in the query".
2) Finding a Suitable Equivalence Notion: Our aim is to implement the high-level encryption scheme defined in Step 1 so that it is distance-preserving. The distance is defined for pairs of data items, but encryption is done item-wise. So we introduce an intermediate notion defined for individual date items: For a given distance measure, an *equivalence notion* captures the characteristics of a single data item that must be preserved upon encryption, cf. Definition 2. For instance, when encrypting graphs (i.e., $\mathcal{S}$ is the set of all valid graphs), a characteristic to be preserved could be the number of vertices.

   Definition 2 (c-Equivalence)**.** *Let $\mathcal{D} \subseteq \mathcal{S}$ be a data set and* $\mathrm{c} : \mathcal{S} \to \mathcal{S}$ *a function (*characteristic*). In addition, let* $\mathtt{Enc}$ *be an encryption algorithm for data items in $\mathcal{S}$. Then* $\mathtt{Enc}$ *ensures c-equivalence if*

   $$\forall x \in \mathcal{D} : \mathtt{Enc}(c(x)) = c(\mathtt{Enc}(x)).$$

3) Ensuring the Equivalence Notions: In this step, one has to implement the high-level encryption scheme defined in Step 1 so that it ensures the equivalence notion from Step 2. To this end, we deploy property-preserving encryption classes introduced in Section II.
4) Security Assessment: If one uses only schemes whose security is known from the literature, the security assessment is given; this is the desired case. Otherwise, a security analysis as, e.g., in [13] is needed.

IV. Distance-Preserving Encryption for SQL Logs

In this section, we instantiate our Kit-DPE procedure with SQL query logs and four well-known distance measures.

A. Security Model

As first step, we have to specify the threat model, i.e., the attacks to shield against, and a high-level encryption scheme for SQL logs. We specify them as follows.

1) Threat Model: As stated in Section II-1, one has to shield against passive attacks only. Hence, it is necessary to instantiate the abstract passive attacks stated in Section II-1 for query logs. Literature already features this [9]. See Example 3.

Example 3 (Cipher-Text Only Attack → Query-Only Attack [9])**.** *In a query-only attack, the attacker only has access to the encrypted query log and tries to infer the plain-text values of constants, relation names as well as attribute names of a given encrypted query.*

2) High-Level Encryption Scheme: Intuitively, SQL queries can be encrypted in various ways, for instance by encrypting the query string as a whole. However, if we want to hide the names of relations, attributes and values of the attributes in the database only, it is sufficient to encrypt only these parts of the queries with different encryption functions, cf. Example 4. The Tuple $(\mathtt{Enc}^{Rel}, \mathtt{Enc}^{Attr}, \{\mathtt{Enc}^{A.Const} : \text{Attribute } A\})$ forms our high-level encryption scheme. Here, $\mathtt{Enc}^{Rel}$ is an encryption algorithm for relation names, $\mathtt{Enc}^{Attr}$ one for attribute names and $\mathtt{Enc}^{A.Const}$ one for constants belonging to an Attribute A. While the idea of encrypting SQL queries in this way is not new [14], it has not been studied so far how to instantiate it so that it is distance-preserving.

Example 4. *For* $Q =$ '$\mathtt{SELECT}\ A1\ \mathtt{FROM}\ R\ \mathtt{WHERE}\ A2 > 5$' *the encrypted query is*

$$\begin{aligned}\mathtt{Enc}(Q) = {}&\text{'}\mathtt{SELECT}\ \mathtt{Enc}^{Attr}(A1)\ \ \mathtt{FROM}\ \mathtt{Enc}^{Rel}(R)\\ &\mathtt{WHERE}\ \mathtt{Enc}^{Attr}(A2) > \mathtt{Enc}^{A2.Const}(5)\text{'}.\end{aligned}$$

B. Equivalence Notions for SQL Query Distance Measures

The second step of KIT-DPE is defining an equivalence notion, i.e., the Function c defining the characteristics of a query to be preserved (cf. Definition 2), for each distance measure considered. Table I is an overview of query-distance measures from literature and of the results of this and the next section. To compute two of the measures, i.e., query-result and query-access-area distance, it is not sufficient to only share the query log. For instance, to calculate query-result distance, the content of the database is needed as well (cf. Table I). We now introduce the measures and the characteristics to be preserved.

1) Token-Based Query-String Distance: For token-based query-string distance, one interprets an SQL query as a set of tokens and uses a set-distance measure like the Jaccard measure to calculate the distance, cf. Definition 3.

Definition 3 (Token-Based Query-String Distance)**.** *Let* $Q1$*,* $Q2$ *be SQL queries. Then the token-based query-string distance between* $Q1$ *and* $Q2$ *is*

$$d_{Token}(Q1, Q2) = 1 - \frac{|\text{tokens}(Q1) \cap \text{tokens}(Q2)|}{|\text{tokens}(Q1) \cup \text{tokens}(Q2)|}.$$

For token-based query-string distance, the characteristic to be preserved is the token set of the queries, i.e., $c = tokens$. We dub this equivalence notion *token Equivalence*.

2) Query-Structure Distance: In [15], the authors extract semantically important features from a Query Q, dubbed *features*(Q). A feature of a query is a tuple representing a part of its structure. See Example 5 for examples of features.

Example 5. *Consider* Q *from Example 4. Its feature set is* $features(Q) = \{(\mathtt{SELECT}, A1), (\mathtt{FROM}, R), (\mathtt{WHERE}, A2 >)\}$.

Query-structure distance is the distance of the feature sets of two queries with the Jaccard measure. For this distance, the characteristic to be preserved is the feature set of the queries. We name this equivalence notion *structural Equivalence*.

3) Query-Result Distance: Query-result distance is the Jaccard distance of the tuples in the results of the queries. Note that the result of a query depends on the state of the database. Thus, besides the query log itself, it is necessary to share the content of all attributes accessed by at least one query in the log of the *encrypted* database as well (DB-Content in Table I). For query-result distance, the characteristic to be preserved is the set of result tuples of the query, cf. Definition 4.

Definition 4 (Result Equivalence)**.** *Let* Q *be a query. Then* $\mathtt{Enc}$ *ensures result equivalence for* Q *if*

$$\mathtt{Enc}(result\_tuples(Q)) = result\_tuples(\mathtt{Enc}(Q)).$$

4) Query-Access-Area Distance: Query-access-area distance is based on the overlap of access areas. The access area of a Query Q regarding an Attribute A, $acccess_A(Q)$, is the part of the domain of A accessed by Q [16]. Using this notion, we define access-area distance in Definition 5.

Definition 5 (Query-Access-Area Distance)**.** *Let* $Q1, Q2$ *be SQL queries and* $Attr_{Q1,Q2}$ *the set of attributes accessed by* $Q1$ *or* $Q2$*. Then the access-area distance of* $Q1$ *and* $Q2$ *is*

$$d_{AE}(Q1, Q2) = \frac{1}{|Attr_{Q1,Q2}|} \cdot \sum_{A \in Attr_{Q1,Q2}} \delta_A(Q1, Q2)$$

where

$$\delta_A(Q1, Q2) = \begin{cases} 0 \text{ if } acccess_A(Q1) = acccess_A(Q2) \\ x \text{ if } acccess_A(Q1) \cap acccess_A(Q2) \neq \emptyset \\ 1 \text{ otherwise} \end{cases}$$

for $x \in (0, 1)$ *with a default value of* 0.5.

The corresponding equivalence notion is *access-area equivalence*, where $c = access_A$ for all Attributes A is preserved.

C. Ensuring Equivalence Notions

The third step of the KIT-DPE procedure is selecting a combination of property-preserving encryption schemes ensuring the equivalence notions just defined. Generally, several encryption classes ensure an equivalence notion. For instance, for token equivalence, we can choose the identity function as "encryption algorithm" providing no security, or some deterministic scheme with higher security level. So we always select the schemes from an encryption class according to Definition 6 using the encryption-class taxonomy from Figure 1.

Definition 6 (Appropriate Encryption Class)**.** *For a given equivalence notion and encryption algorithm in (*Enc^{Attr}*,* Enc^{Rel}*,* $\{\mathtt{Enc}^{A.Const} : \text{Attribute } A\}$*), an encryption class is appropriate according to an encryption-class taxonomy if (1) it ensures the equivalence notion and (2) provides the highest possible security.*

TABLE I
OVERVIEW OF QUERY-DISTANCE MEASURES.

Distance Measure	Shared Information			Equivalence Notion	c	$\mathtt{Enc}^{Rel}$	$\mathtt{Enc}^{Attr}$	$\mathtt{Enc}^{A.Const}$
	Log	DB-Content	Domains					
Token-Based Query-String Distance	✓	✗	✗	Token Equivalence	*tokens*	DET	DET	DET
Query-Structure Distance	✓	✗	✗	Structural Equivalence	*features*	DET	DET	PROB
Query-Result Distance	✓	✓	✗	Result Equivalence	*result_tuples*	DET	DET	via CryptDB [8]
Query-Access-Area Distance	✓	✗	✓	Access-Area Equivalence	access_A	DET	DET	via CryptDB, except HOM

Table I lists the appropriate schemes and classes for all SQL query distance measures, indicating that one can find appropriate classes for all distance measures. For two measures, we can rely on CryptDB, an approach for executing SQL queries over encrypted databases reducing implementation overhead.

Observe a specificity of access-area equivalence: As the `SELECT`-clause does not have any influence on the access area of a query, for access-area distance, we can encrypt the values of attributes contained in an arithmetic aggregate function in the `SELECT` clause just with a PROB scheme. This yields a higher security level than one for result distance that uses CryptDB as it is.

D. Security Assessment

The last step of KIT-DPE is the security assessment of the appropriate encryption schemes. As we have used property-preserving encryption schemes known from literature, their level of security is known. Thus, we can reduce the security of the schemes to the security of the encryption schemes for SQL queries we have just specified [9]. This is intended – executing a full security analysis for organizations which want to outsource data analysis is practically impossible.

V. CONCLUSIONS

Many organizations have data that contains valuable information, but cannot analyze it themselves. Therefore, they want to outsource the analysis to a service provider. To ensure confidentiality, they are willing to transfer their data only if it is encrypted. To this end, it is important that the encryption preserves the mining results. Thereby, we introduce distance-preserving encryption (DPE) and propose the KIT-DPE procedure. It establishes how to design a DPE-scheme for arbitrary data and distance measures. We exemplary instantiate this procedure for SQL query logs. In the respective study, we find appropriate DPE-schemes for all distance measures for SQL queries from literature. We use well-known property-preserving encryption classes to implement the DPE-schemes and assess their security. This is different from approaches supporting ad-hoc queries like CryptDB [8] and gives way to higher security. Furthermore, for instance, result equivalence for SQL queries, is also useful for association-rule mining over encrypted SQL logs [17]. Studying the applicability of equivalence notions in different contexts also offers interesting opportunities for future work.

ACKNOWLEDGEMENT

This work was supported by the German Research Foundation (DFG; reference number BO 2129/13-1) and the Federal Ministry of Education and Research (BMBF; reference number 02K15A024).

REFERENCES

[1] O. Pandey and Y. Rouselakis, "Property-Preserving Symmetric Encryption," in *EUROCRYPT*. Springer, 2012.

[2] R. Agrawal *et al.*, "Order-Preserving Encryption for Numeric Data," in *SIGMOD*. ACM, 2004.

[3] D. Defays, "An Efficient Algorithm for a Complete Link Method," *The Computer Journal*, 1977.

[4] M. Ester *et al.*, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases With Noise," in *KDD*. AAAI Press, 1996.

[5] H.-S. Park and C.-H. Jun, "A Simple and Fast Algorithm for K-Medoids Clustering," *Expert Systems with Applications*, 2009.

[6] E. M. Knorr *et al.*, "Distance-Based Outliers: Algorithms and Applications," *VLDBJ*, 2000.

[7] N. Arzamasova, M. Schäler, and K. Böhm, "Cleaning Antipatterns in an SQL Query Log," *TKDE*, 2018.

[8] R. Popa *et al.*, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," in *SOSP*. ACM, 2011.

[9] T. Sanamrad and D. Kossmann, "Query Log Attack on Encrypted Databases," in *SDM Workshop*. Springer, 2013.

[10] J. Li *et al.*, "L-EncDB: A Lightweight Framework for Privacy-Preserving Data Queries in Cloud Computing," *Knowledge-Based Systems*, 2015.

[11] C. Fontaine and F. Galand, "A Survey of Homomorphic Encryption for Nonspecialists," *EURASIP*, 2007.

[12] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-the Advanced Encryption Standard*. Springer, 2013.

[13] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions," in *CRYPTO*. Springer, 2011.

[14] D. Purnamasari *et al.*, "Query Rewriting and Corpus of Semantic Similarity as Encryption Method for Documents in Indonesian Language," in *ICESTI*. Springer, 2016.

[15] N. Khoussainova *et al.*, "SnipSuggest: Context-Aware Autocompletion for SQL," *PVLDB*, 2010.

[16] H. Nguyen *et al.*, "Identifying User Interests Within the Data Space - A Case Study with SkyServer," in *EDBT*, 2015.

[17] J. Aligon *et al.*, "Mining Preferences from OLAP Query Logs for Proactive Personalization," in *ADBIS*. Springer, 2011.