Secure Normal Form: Mediation Among Cross Cryptographic Leakages in Encrypted Databases

Shufan Zhang* Xi He* Ashish Kundu† Sharad Mehrotra‡ Shantanu Sharma§
*University of Waterloo †Cisco Research ‡University of California, Irvine §New Jersey Institute of Technology
{shufan.zhang,xi.he}@uwaterloo.ca, ashkundu@cisco.com, sharad@ics.uci.edu, shantanu.sharma@njit.edu

***Abstract*—Existing secure data outsourcing systems offer users ways to select from different cryptographic primitives supported by the system to encrypt their data to strike a balance between data confidentiality and query performance. Though prior work have identified the danger of mixing cryptographic primitives, they fall short of providing a systematic approach to guide users to prevent such cross-cryptographic leakages. Inspired by the database design theory, we envision *Secure Normal Form*, a new approach to normalize encrypted databases such that the leakages of the partitioned databases are limited to the users' specifications. In this work, we propose a new architecture to support secure normal form. This system includes several new components for secure data outsourcing: (i) an inference mechanism that reasons about additional leakages from weaker encryption techniques, based on semantic data properties (e.g., dependence between attribute values); (ii) a normalization mechanism that converts relational data into secure normal forms, so that the information leaked by the representation is limited to that specified by the user; and (iii) a secure query execution approach over encrypted data in secure normal forms. Our initial experimental results validate the performance improvement over naïve baseline and show that a careful data representation can be allowed without compromising security. We believe that our paper opens a new direction in secure data management.**

***Index Terms*—Secure Data Management, Encryption, Holistic Leakage Accounting, Inference Control.**

I. Introduction

"A chain is only as strong as its weakest link."

Thomas Reid, *Essays on the Intellectual Powers of Man*
Applicable to the security of data management systems.

A Retrospective of Secure Data Management: Twenty-Five Years Later, Where Are We? The growing demand to store and manage large (and increasing) amounts of private data over the cloud has spawned an era of secure data management. Nearly twenty-five years have passed since the first initiative on outsourcing encrypted database as a service [1] and the first cryptographic technique that allows search on encrypted data [2]. System researchers, cryptographers, and practitioners have made significant progress in developing many new algorithms or protocols for storing, transferring, and operating on encrypted data. These advances include several encryption techniques, e.g., deterministic encryption (DET) [3]–[5] for equi-join queries [6], order-revealing encryption (ORE) [7], [8] for range queries, homomorphic encryption [9]–[13] for performing arbitrary computation, non-deterministic encryption (NDET) [14], multi-party computation [15], [16], just to name a few. Hardware-oriented solutions, such as secure enclave [17], [18], have been explored as well to address the same problem. All such existing approaches exhibit trade-offs among the nature of queries that can be executed, the efficiency of query performance, and the security offered to the end users.

The emerging wisdom in the field is that no single approach offers a silver bullet. As a result, diverse secure data management systems have been built, adopting different encryption technologies based on the discrepant use cases — needs for efficiency, security, and the required database operators. These include systems such as CryptDB [6], Seabed [19], HE3DB [20], SDB [21], Jana [22], Microsoft Always Encrypted [23], AWS Clean Room [24], and MongoDB Querable Encryption [25]. Some systems focus on implementing a single type of cryptographic technique, but they support only limited queries efficiently [23], [25]. Others support a more general class of queries by applying multiple cryptographic techniques each offering different security guarantees to meet the performance requirements of a data management system in practice. For example, CryptDB [6] and Seabed [19] allow users to encrypt different parts of data using different encryption mechanisms. Users can perform a "sensitivity analysis" [26] on the schema specification to determine the encryption schemes — highly sensitive columns can be encoded with a strong encryption technique, such as the Advanced Encryption Standard (AES) that ensures full confidentiality or a fully homomorphic technique in case computation (e.g., aggregation) may need to be performed on the field, while less sensitive attributes are encrypted with weak encryption like DET or ORE to make query processing more efficient. By allowing users to encrypt different parts of the data using different cryptographic techniques, such systems empower users to explore tradeoffs between security and performance. Users can choose a representation that delivers the best performance subject to the security guarantee that the adversary, often the cloud server, should not learn anything useful about the plaintext, other than the necessary leakages specified by the weak(er) encryption technique used.

Cross-Cryptographic Leakages. The final security offered by systems that use multiple cryptographic techniques to encrypt different parts of the database is less well understood. Prior work [27]–[30] primarily analyzes the information leakage within a single weakly encrypted database column. For example, when employing DET on a column, the ciphertext's distribution may reveal the true value of the entire column

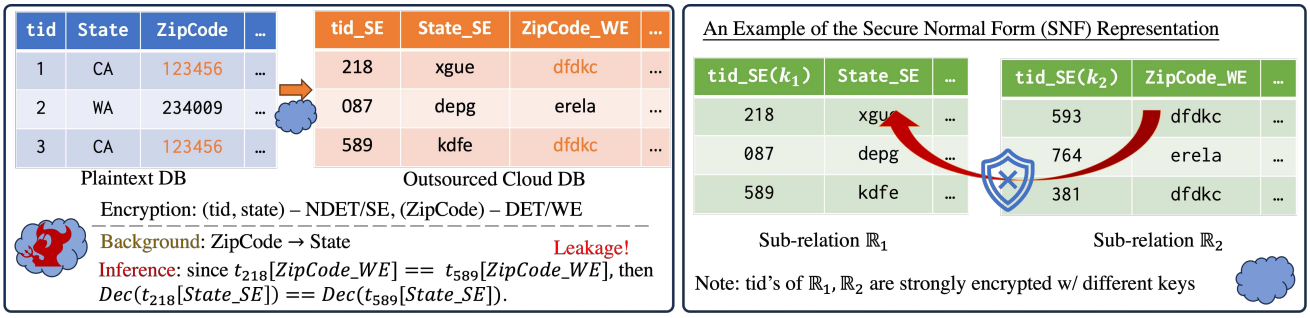

Fig. 1: An example of lacking holistic leakage accounting in secure data management: (Left:) leakages can happen when multiple encryption techniques are used in outsourced databases; (Right:) SNF Representation can avoid such leakages.

based on auxiliary knowledge on the data distribution [27]. Note such leakages may be permitted in practice for better performance, but they can propagate to affect other parts of the dataset, giving rise to unintended leakages, what we term *cross-cryptographic leakages*. We illustrate such a leakage in a simple example below.

Example 1. (Leakage Across DET and NDET). Consider a database relation that contains two or more attributes, e.g., tid, State and ZipCode, and consider three tuples in this relation; see the left part of Figure 1. Suppose the data owner encrypts tid (tuple id) and State columns with NDET, and ZipCodes with DET to enable the equality test. Due to DET, the distribution of ZipCode column is revealed. Although NDET itself reveals nothing about the State column, ZipCode and State columns are functionally dependent (i.e., ZipCode $\rightarrow$ State). A semi-honest cloud server can thus learn more than what it is allowed about State data. That is, as t_{218}[ZipCode_WE] $=$ t_{589}[ZipCode_WE], the cloud server can infer the plaintext $Dec(t_{218}$[State_SE]$) = Dec(t_{589}$[State_SE]$)$. Here, WE refers to weak encryption techniques, such as DET that reveals data distribution and ORE that reveals ordering of values. SE refers to strong encryption techniques, such as NDET that does not reveal anything. □

This toy example illustrates how functional dependency between columns can lead to additional leakage. Prior work such as [27], [31] have considered more subtle inference attacks, showing that when systems use weak encryption techniques general correlations between columns can propagate the leakages from a weakly encrypted column to other columns. In particular, prior work [27] shows that if a relation represents an attribute using DET and another attribute using ORE, it may reveal the entire tuple based on background knowledge. [31] further shows that based on background knowledge and property revealing encryption (e.g., ORE) on a column of a table can reveal the data of another strongly encrypted column encrypted of the table, if the columns are dependent. Such prior work identify the challenges in mixing multiple cryptographic approaches. However, they fall short of providing a path forward wherein diverse cryptographic techniques with different levels of security can be composed while still ensuring provable security guarantees.

Despite negative results in [27], [31], we would like to devise a way forward to build provably secure cross-cryptographic solutions that supports weaker cryptography without any unintended cross-cryptographic leakage. Such an approach would offer end-users ways to explore tradeoffs between provable security and performance empowering them to make informed choices. This paper offers the first step in developing the theoretical underpinning to understand how leakages can spread when multiple cryptographic primitives are used and system designs that reduce/prevent unintended leakages that may result when multiple primitives are used.

Research Questions. We articulate our goals in the form of the following two research questions (RQs) that are necessary to answer in building toward a secure cross-cryptographic solution.

Preventing Cross-Cryptographic Leakages

*RQ1. Is there a **systematic approach** to account for cross-cryptographic leakages in encrypted databases?*

*RQ2. How can we build an **efficient system** to holistically mitigate these leakages, thereby users can use secure data management services carefree?*

The first RQ aims to detect unintended cross-cryptographic leakages within an encrypted database. Addressing it requires us to explore theoretical principles to reason about all implicit /explicit leakages that may occur given a relational data representation that uses multiple cryptographic primitives to encrypt different parts of the data. Given a way to reason about leakages, the second RQ seeks solutions to prevent unintended leakages[1] while minimizing performance overheads. Next, we will discuss the core idea for constructing our solution to these two RQs.

Towards A Solution. To address RQ1 and RQ2, we take inspiration from the relational design theory that has so successfully led to developing a formal basis of reasoning about redundancy

[1]That is, leakages other than the permissible ones the user explicitly defined by choosing a weaker cryptographic primitive that may reveal some property of data.

Our Vision: Secure Normal Form (SNF)

Normalizing the encrypted database, where the leakages of each partition are restricted to only what a user explicitly allows (by specifying encryptions).

in relational representations. Similar to a framework to reason about database constraints (e.g., functional dependencies), we envision a theoretical framework that given data semantics in the form of constraints such as correlations, and cryptographic primitives used to encrypt data, allows us to reason about leakages that may ensue from the data representation. Once such a theoretical framework has been established, just like relational normalization that transforms a given representation to an equivalent representation without redundancy by partitioning relations into sub-relations, we envision transforming encrypted representations into equivalent encrypted relational representations such that the transformed representation prevents unintended leakages[2].

Secure Normal Form. We refer to such a transformed representation (where leakages are restricted to only those permitted directly by the choice of encryption technique specified by user, and no other unintended leakage from inference) as being in a **secure normal form** (SNF). That any relational representation can be transformed into an equivalent SNF can be easily seen by revisiting Example 1 which had an unintended leakage about two ciphertext representations of the State field representing the same plaintext through the DET representation of the ZipCode field coupled with the functional dependency between ZipCode and State fields. Consider a different representation wherein we partition the table into two sub-relations, one containing ZipCode and another State with a new attribute identifier, strongly encrypted using non-deterministic encryption to link the appropriate rows in the two sub-relations. The original relation can be reconstructed by performing a join over the identifier — of course, such a join operator must be implemented carefully to prevent adversary to learn correspondence between the tuples in the two sub-relations. A partitoned representation such as above prevents the leakage resulting from the DET encryption of ZipCode to spread to other attributes and can be argued to be in SNF with no unintended leakages. In general, for any relation with some attributes encrypted using weak encryption techniques such as ORE or DET, or cryptographic protocols over attributes that may leak data, we can always transform the relation into SNF by simply partitioning the table to store each column separately with appropriate secure mechanism to reconstruct the tuples by joining them using an identifier.

While such an approach will generate SNF, as we will see, several SNF representations can exist for a given relation, some more preferable than others (we will formally define a criterion of how to rank representations in Section II). Our goal, thus, becomes that of developing algorithms to transform data representations that may contain unintended leakages into desirable representations that are in secure normal form.

To build our approach to transforming encrypted representations into SNF, similar to relational design theory, we seek to define a *sound and complete inference mechanism* to reason with leakages when data is converted into proper server-side representations based on which partitioning algorithms for SNF can be built. Such algorithms, in turn, can guide the development of data outsourcing algorithms using which data owners can explore tradeoffs between security and performance, serving as a risk management and mitigation strategy for outsourcing relational data.

In addition to leakage from ciphertext and corresponding implementation of relational operators using the cryptographic primitives, additional leakages can arise when queries are executed over a normalized encrypted database when cross-cryptographic primitives are used. To avoid such query-time leakages, we need to support the encrypted database with a new secure query processing schemes.

Roadmap. In this paper, in addition to achieving representations that are in SNF, we present a forward-looking system architecture for query execution that is secure, as well as our early results, and the opportunities and challenges toward a full realization of our vision. We start with an analysis to obtain a better understanding of leakages (Section II), and present our proposed system architecture that allows encrypted data representation based on secure normal forms (Section III). The paper focuses on the following three aspects:

- *Efficient Leakage Reasoning.* An efficient inference engine to automatically reason about additional leakages from the representation of the outsourcing database.
- *Efficient Normalization.* A normalization mechanism with heuristics that can efficiently find a partitioned representation in secure normal form, while achieving desired optimization objectives.
- *Secure Query Processing.* A new query processing mechanism, deployed on the server side, that can securely execute queries across sub-relations in the secure normal form representation.

In addition, we explore multiple dimensions to extend the solution space for constructing SNFs (Section IV). Our preliminary experimental results show promising evidence in realizing our vision of provably secure representations that exploit multiple cryptographic techniques simultaneously. We discuss directions, opportunities, and challenges of building a full-fledged system based on our vision in Section V. Finally, we offer concluding remarks in Section VI. As future work, we would like to complete/implement a prototype for a system that supports data representation based on SNFs and integrate such an approach with industrial cloud databases such as Cisco's Panoptica [35]. We believe, that once realized, the SNF vision can usher in a novel design paradigm for secure data management and reshape researchers' perspectives

[2]Partitioning has also been studied in distributed system settings [32]–[34] and serves different security or deduplication goals.

within the community to address cryptographic leakages from a more comprehensive and systematic standpoint. Hopefully, this paper can inspire people to take the opportunity to solve the open challenges with realizing systems that support data representations based on SNF. These challenges include but are not limited to, the need for a more precise characterization of leakages, extending SNFs to prevent leakages from other system components like logging or indexing, the integration of probabilistic modeling into SNFs, and the development of frameworks capable of sustaining SNFs across dynamically growing databases or other database types beyond the traditional relational model.

II. Secure Normal Form

In this section, we formally define the notion secure normal form and discuss criteria for selecting amongst multiple SNF representation of the encrypted relation. We begin by first discussing the notion leakage including concepts of permissible and unintended leakages based on which SNF are defined. In discussing leakage and SNF, we will consider a setup in which a *data owner* outsources the database to a *public cloud*. The data owner specifies different encryption techniques that are used to protect each attribute while enabling functionalities to execute queries efficiently over the outsourced encrypted database. We assume the cloud is a semi-honest adversary, hosting the encrypted data and executing user-submitted queries faithfully, but the cloud tries to gain information about the data from both encrypted data at rest and during query execution.

A. Information Leakage

Leakages can be viewed as the information about *targeted data objects* that the adversary gains from seeing the ciphertext, which is encrypted with a specific technique. An example of a targeted data object could be a cell in a relation. Such leakages that happen when data is stored at rest due to the choice of encryption technique used. It could also happen when queries are executed based on the access patterns, intermediate computations, query patterns, output size – any observation an adversary can make from the execution of queries. We refer to the former as ***static leakages*** and the latter as ***dynamic leakages***. SNF definition deals with leakage from ciphertext and so we focus for now on static leakages and formally define it as follows:[3]

*Definition 1 (**Leakage**):* We say there is a leakage on a collection of plaintext C from its ciphertext representation $\mathbb{C}$, iff, $\forall c_i \in \mathbb{C}$, such that,

$$\left| \Pr_{\mathcal{A}}[\mathrm{Dec}(c_i) = v \mid \mathbb{C}, b] - \Pr_{\mathcal{A}}[\mathrm{Dec}(c_i) = v \mid b] \right| > \mathsf{negl}(1^{\eta}),$$

where $\mathrm{Dec}(c_i)$ refers to the plaintext corresponding to the ciphertext c_i, v is any plaintext value c_i can take, b denotes the background knowledge that an adversary $\mathcal{A}$ can obtain. ■

[3]Dynamic leakages can be equivalently defined by conditioning the adversarial knowledge not just on the knowledge of ciphertext but other query-specific leakages such as access patterns, output volume, etc. See [36] for such an extended definition of leakages.

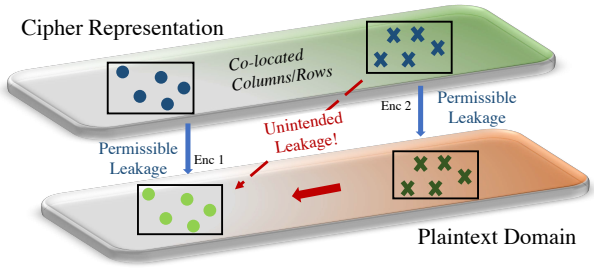

Fig. 2: An illustration of permissible and unintended leakages, and when unintended leakages can happen at the representation level.

This definition says a leakage happens when an adversary's probability of successfully guessing the plaintext value of a data object increases by a non-negligible amount (w.r.t. a security parameter η) with the knowledge of the ciphertext as compared to only the background knowledge. We further categorize leakages as ***permissible*** due to the nature of cryptographic techniques used to enable database functionalities, and ***unintended*** due to the dependencies among data.

Permissible Leakages $\mathcal{L}_P$. Let R be a relational scheme with a set of *permissible leakages*, denoted by $\mathcal{L}_P$. Let $\mathbb{R}$ be its corresponding representation with (possibly multiple) cryptographic techniques used to encrypt data in R and store it at the server[4]. Permissible leakages $\mathcal{L}_P$ refer to the leakages that the data owner is willing to accept. In our setup these correspond to explicit leakages that result due to the choice of cryptographic techniques the database owner has chosen and are a result of the property revealing nature of the chosen primitives.

Example 2. In Example 1, the user specifies to use DET for the ZipCode column. Leakages on the ZipCode data distribution through DET are permissible leakages. □

Unintended Leakages $\mathcal{L}_U$. Unintended leakages, denoted by $\mathcal{L}_U$, are those leakages not specified by users but can be inferred based on permissible leakages.

Example 3. In Example 1, due to the combined use of NDET and DET, there is unintended leakage on the State column via functional dependencies. This leakage has not been defined by the user and is an unintended leakage. □

Figure 2 further illustrates permissible and unintended leakages, where unintended leakages can occur at the representation level. The upper plane in the figure illustrates a cipher representation that stores different parts/attribute values of the data (dots and crosses). The lower plane is the underlying plaintext data of the corresponding ciphertext. Permissible leakages (blue arrows) of plaintext based on the corresponding ciphertext are a property of the encryption schemes used

[4]In general, $\mathcal{R}$ may store some data at the local machines as well if that storage scheme helps prevent leakage, though such a representation does not fundamentally change the problem we study.

("Enc 1" and "Enc 2"). The solid red arrow in the plaintext domain denotes the correlation or dependencies between two parts of the data, which leads to unintended leakages (dotted red arrow) from one part of the ciphertexts to the other part of the data.

B. Secure Normal Form

We can now provide a more formal definition of *secure normal form (SNF)* representation of a relation.

*Definition 2 (**Secure Normal Form**):* Let R be a relation. Let $\mathbb{R} = \{\mathbb{R}_1, \mathbb{R}_2, \ldots, \mathbb{R}_n\}$ be a representation of R. $\mathbb{R}$ is said to be in secure normal form (SNF) with respect to $\mathcal{L}_P$ if and only if the set of leakages that can be inferred from $\mathbb{R}$ about relation R are limited to only those in $\mathcal{L}_P$. ■

The representation $\mathbb{R}$ consists of a set of sub-relations (of the original relation R), where each $\mathbb{R}_i$ contains one or more attributes of R, encrypted using a specific cryptographic technique. For $\mathbb{R}$ to be a SNF representation of R, it must prevent any further leakage except for permissible leakage specified in $\mathcal{L}_P$. Further, $\mathbb{R}$ satisfy the following properties:

- *Sub-relations Unlinkability.* Given any two sub-relations $\mathbb{R}_i, \mathbb{R}_j$ in SNF representation, one cannot link tuples from $\mathbb{R}_i$ with tuples from $\mathbb{R}_j$.
- *Lossless Reconstructability.* The original database can be reconstructed by correctly performing a secure relational operations (e.g., unions, joins, or a combination of the two) over the sub-relations in secure normal forms.

To achieve the above two properties simultaneously, sub-relations in SNF are associated with a tid attribute, encrypted always with strong encryption, with different cryptographic keys. We discuss in the following section how we perform secure reconstruction and query execution over the sub-relations.

Example 4. Referring to the right-side of Figure 1, the outsourced database in secure normal form stores two sub-relations $\mathbb{R}_1$ and $\mathbb{R}_2$. The leakages to the State and Zip-Code columns are restricted to their own partitions. That is, the equality information leaked from ZipCode_WE does not spread to the State column. □

C. Maximally Permissive SNF

SNFs could be more secure (i.e., leak less) than what the user specifies in the schema annotated with encryption techniques. For example, if the user annotates attribute ZipCode to be encrypted with DET, it is allowed to encrypt ZipCode with strong encryption in SNFs, even though it offers more security than what the user asked for. Therefore, it is natural to ask what is a "good" SNF representation among all feasible solutions. A good candidate reduces the query execution overhead that may result due to processing over partitioned relations. As a criterion for choosing a good partitioning, we develop the notion of *maximally permissive* SNFs. a good partitioning.

*Definition 3 (**Maximally Permissive SNFs**):* Let R be a relation, $\mathcal{L}_P$ be the set of permitted leakages on R, and $\mathbb{R} = \mathbb{R}_1, \mathbb{R}_2, \ldots, \mathbb{R}_n$ be its representation that is in SNF. We say $\mathbb{R}$ is maximally permissive, if for any sub-relation $\mathbb{R}_i \in \mathbb{R}$, adding any additional attribute $a \in R$ to $\mathbb{R}_i$, or weakening the cryptographic approach used to represent any attribute in $\mathbb{R}_i$ leads to the relation $\mathbb{R}_i$ not being in SNF. ■

Maximally permissive representations are desirable, since queries that are limited to a set of attributes that are stored together can be implemented more efficiently compared to the queries that span different sub-relations (and hence, require an expensive join). Approaches to generating maximally permissive SNFs will be discussed in Section IV.

III. System Design

In this section, we describe a possible design of a encrypted database that exploits SNF to ensure security while supporting multiple crytographic primitives. The envisioned system outsources data by transforming it into a corresponding secure normal form. Queries are then transformed to execute over the SNF representation. We discuss how the system generates the SNF based data representation, followed by how queries are processed in the system. We leave the discussion at a high level and point out possible directions for realization while deferring other interesting optimizations to Section V.

A. Transforming Relations into SNF

Given a relation R, a set of cryptographic techniques $\mathcal{C}$ over attributes of R (which defines for us the set of permitted leakages $\mathcal{L}_P$), and a specification $\mathcal{D}$ of which attributes/attribute values are independent (and hence cannot lead to leakage through inference) or correlated (and could likely lead to leakage), the system partitions R into a set of subrelations that satisfy the SNF.

To compute SNF, first, a closure of a set of all leakages (both permitted as well as unintended) that the current representation may lead to, is computed (lines 1-2 of Algorithm 1). We denote this closure by $\mathcal{L}^+$ and $\mathcal{L}^+ = \mathcal{L}_P \cup \mathcal{L}_U$. Then, a normalization algorithm is used to partition R (line 3) to create a set of sub-relations $\{R_1, \ldots, R_n\}$ such that the only leakages in the partitioned representation correspond to permissible leakages $\mathcal{L}_P$. Once a set of sub-relations have been determined, a tid field is added to each sub-relation R_i using which we can successfully reconstruct the original relation (thus achieving the reconstructability requirement of SNF). Finally, sub-relations are outsourced after attributes of the relation have been suitably encrypted using cryptographic techniques specified in $\mathcal{C}$. The tid field added to each sub-relation is encrypted using a strong encryption technique that prevents unlinkability between sub-relations. We next discuss several aspects of the above techniques further including how to infer leakages and how to transform a relation into SNF.

Modeling Independence/Dependence in Data. Transforming relation into an SNF requires as input specification of which data/attributes are independent/correlated. Database literature has a rich tradition of modeling dependence in the form of data dependencies, such as functional dependencies, denial constraints, conditional dependencies, aggregation constraints, etc. In addition, probabilistic graphical models have been

extensively used to represent dependence and independence in data. These include use of graphoids to describe conditional independence [37], directed graphs (e.g., Bayesian model [38]) or undirected models (e.g., Markov model [39], [40]). Any of these models, including their combination, could be used in our context of representing data dependence/independence. Furthermore, either data owners could specify data dependencies using such models as part of specifying data semantics, or data dependencies could be learnt/inferred from data [41]. While SNF-based approach for secure data processing works either way, it does require that the specification of data dependence/independence be complete — i.e., for any two data objects, it should be algorithmically determinable if the data items are independent or dependent.

Inferring Leakages. The Step 2 of our approach to transforming data into SNF calls ANALYZELEAKAGECLOSURE that takes as input the knowledge of (in)dependencies, the schema, and the database, and outputs a closure of leakages (i.e., $\mathcal{L}^+ := \mathcal{L}_P \cup \mathcal{L}_U$). Implementing such a function requires careful analysis of the properties of each encryption technique. In particular, given the property revealing aspect of a weaker encryption technique on one attribute, we need to carefully understand how it combines with the data dependencies to reveal semantics of other attributes. For now, we do not differentiate among different types of leakages and make a *conservative* assumption that whenever an attribute is dependent upon another weakly encrypted attribute, the leakage spreads to the dependent attribute as well. Based on this assumption, we only need to know the cryptographic technique for each column and the dependencies between columns to derive the leakage closure. Then, we can parse the schema specified originally to obtain the permissible leakages based on the encryption properties and determine if a given representation contains any unintended leakages. Thus, determining unintended leakages in a given representation reduces to the problem of determining if two columns are independent or not which can be achieved through algorithms discussed above.

Normalization/Partitioning Algorithms. The PARTITIONING component takes as input the database R and a set of leakage rules $\mathcal{L}^+$ obtained from the inference component and returns a normalized database consisting of sub-tables $\{\mathbb{R}_1, \ldots, \mathbb{R}_n\}$ that are in SNFs. A naïve way to implement the normalization algorithm is to use the chase [42] to generate every possible partitioning and iterate over them by calling ANALYZELEAKAGECLOSURE function to determine the ones in SNF (i.e., if for all sub-relation in a partition, $\mathcal{L}^+ = \mathcal{L}_P$, then this partition is in SNF). However, this approach incurs an exponential number of cases. We will design heuristics for partitioning, based on the input leakage closure $\mathcal{L}^+$ in Section IV. Note partitioning a database into SNF might, highly possibly, not be unique. Among all candidates, we select *maximally permissive SNFs* (see Definition 3), as our outputs of partitioning.

B. Executing Queries over SNF

We now turn our attention to executing secure query execution when data is represented using SNF. Let $q_i(A_i, \ldots, A_j)$ be the query that the semi-honest cloud server receives from a client, where A_i refers to an attribute. If all the attributes referred to in the query $q_i(A_i, \ldots, A_j)$ occur in a single sub-relation, the system can directly execute the query. This is the same situation as in existing encrypted database systems wherein all fields in the query are collocated in the same relation (lines 6-7). In contrast, if the attributes in the query $q_i(A_i, \ldots, A_j)$ do not exist in a single sub-relation, the system may need to access fields across multiple sub-relations $\mathbb{R}_i, \ldots, \mathbb{R}_j$ (that together cover all attributes in the query) and join the records in those sub-relations (lines 9-10) using the tid column to reconstruct the necessary parts of the partitioned records. After such a reconstruction the original query can be executed and answers are returned to the user (line 11). As mentioned in Section II, query execution can lead to leakage by an adversary observing the execution of the query. The *query-time leakages* or *dynamic leakages* can occur and reveal which ciphertext is accessed during query execution (i.e., access pattern attacks [43]) and/or observing the volume of intermediate or final outputs produced [44]. Such leakages coupled with background knowledge often can result in semantic leakage. In the context of SNF representation, when the query requires data stored in more than one sub-relation to be accessed, unless the reconstruction is done carefully, adversary may gain knowledge of the correspondence between records stored across different sub-relations, leading to leakage. To ensure that no leakage occurs due to SNF representation, we need to prevent linkability among sub-relations from being leaked during reconstruction. We, thus, need to perform the reconstruction carefully to ensure that the adversary does not gain insight on which tid values across two sub-relations are equal. We can protect against adversary learning about linkability in one of the following two ways:

Oblivious Reconstruction. We could use secure hardware such

Algorithm 1: System Overview

Input: Database $R(tid, A_1, A_2, \ldots, A_n)$, A set of cryptographic techniques $\mathcal{C}$ specified over the schema of R

```
/* Data owner performs locally:                          */
D ← DEPENDENCYINFERENCE(R)
L+ ← ANALYZELEAKAGECLOSURE(R, C, D)
{R_1, ..., R_n} ← PARTITIONING(R, L+)
SNF := {R_1, ..., R_n} ← ENCRYPTION({R_1, ...}, C)
/* Outsourcing SNF to the cloud, and the cloud
   executes the following:                            */
Receive query q_i(A_i, ..., A_j)
if {A_i, ..., A_j} ⊆ R_t then
    return QUERYANSWERING(q_i, R_t)
else
    {R_i, ..., R_j} ← QUERYMATCHING(SNF, q_i)
    R ← OBLIVIOUSJOIN({R_i, ..., R_j})
    return QUERYANSWERING(q_i, R)
end
```

as SGX [17], [18] in conjunction with oblivious random access memory (ORAM) [45], [46] to reconstruct the original relation from sub-relations. In particular, when tuples matching a user's query (e.g., selection on a particular attribute) have been determined over a given sub-relation, the corresponding tid of the relation can be decrypted in the secure hardware and ORAM used to retrieve the corresponding parts of the tuple stored in other sub-relations. Since ORAM techniques such as [15], [36], [47]–[49] prevent the adversary from learning the access pattern, linkability of records across sub-relations will be prevented.

Query Binning. We could also prevent linkability among tables by using a query binning technique [50] developed in the context of secure joint query processing over partially sensitive data where sensitive and non-sensitive data was partitioned into separate sub-relations. Retrieving a query, say a selection query (e.g., a predicate such as $A = 5$) separately on sensitive (stored in an encrypted representation) and non-sensitive data (stored in plaintext) would clearly break the scheme since the adversary would immediately learn which encryped tuples refer to the plaintext in the query. To protect against such a leakage, [50] explored a binning approach that carefully retrieves data from both sensitive and non-sensitive tables to prevent the adversary from gaining knowledge of such correspondences. Depending upon the encryption technique used for sensitive data, the percentage of data that is sensitive versus which is not, [50] showed significant advantages in terms of performance of the binning approach compared to techniques such as Path-ORAM or oblivious computation. The technique of binning, while designed to support secure computation when data is partially sensitive, can nevertheless, be generalized to our setting when reconstructing tuples from different sub-relations.

Note that above, we have only discussed potential approaches to reconstruct records when data is stored in SNF. How the reconstruction operation, executed obliviously, can be integrated into a full-fledged query execution to prevent dynamic leakage from the overall query execution remains an important challenge going forward.

IV. Solution Space for Constructing SNFs

The core of SNF is the normalization algorithm. A database relation can be partitioned vertically or horizontally, or via a mixture of both. We first restrict our sight to only vertical partitioning to describe a few possible normalization algorithms in this scope. We then describe the possibilities in considering horizontal partitioning also to broaden the solution space for better optimizations.

A. Preliminary Discussions on Partitioning Strategies

Given R and $\mathcal{L}^+$ on R, our goal is to transform R into a representation $\mathbb{R}$ such that $\mathbb{R}$ is in SNF w.r.t. $\mathcal{L}_P$. To do so, we perform partitioning over the relation R.

Vertical Partitioning. We say $\mathrm{vp}(R) = \{R_1, R_2, \ldots, R_k\}$ is a vertical partition of a relation R, if:

- $\forall i \in [k] : \mathrm{attr}(R_i) \subseteq \mathrm{attr}(R)$, *and*
- $R = R_1 \bowtie \ldots \bowtie R_k$.

Each sub-relation with vertical partitioning contains the *tid* attribute strongly encrypted with different cryptographic keys. This ensures that we can reconstruct the original relation R from the sub-relations, while the linkability among sub-relations will not leak when data is stored at rest. Note that there is always a trivial algorithm for vertical partitioning, as follows:

Trivial Strategy. We represent $R(A_1, \ldots, A_n)$ as a set of n sub-relations: $\mathbb{R}(tid\_SE(k_i), \mathbb{A}_i), \forall i \in [n]$, where k_i is the encryption key. Each attribute $\mathbb{A}_i$ is encrypted using an appropriate cryptographic technique that is specified in the pool of specified cryptographic techniques $\mathcal{C}$.

While generating SNF for a given relation is simple with the trivial strategy, our objective, as discussed in the previous section, is to support a *maximally permissive property*. The two naïve approaches – encrypting all attributes of R using strong encryption, or over-split R into sub-relations with a single attribute from R – both are in SNF, but neither might be maximally permissive. We consider the following two normalization strategies, which are based on variants of the *hill climbing* heuristic.

Strategy 1: Non-Repeating. We start with an unintended leakage or a dependency that causes a leakage. We put the corresponding attributes in the dependency into separate sub-relations. For the rest of the attributes in the schema, we take each attribute and iterate over the leakages closure. The goal is to identify a candidate set of existing sub-relations for this attribute. A sub-relation is put into this candidate set, if the encrypted attribute does not incur unintended leakages with existing attributes in this sub-relation. If the candidate set is not empty, we add the attribute into *any* of the sub-relations in it; otherwise, we create a new sub-relation to place this attribute. This process will end when every attribute in the schema is properly allocated to the sub-relations.

Strategy 2: Max-Repeating. This strategy is a variant of the previous one, where the only difference is that each attribute will be put into *every* sub-relation in the candidate set, such that every sub-relation will be in SNF.

Note that the maximal permissiveness does not specify how many times an attribute from R can repeat in $\mathbb{R}$ in SNF. The two aforementioned strategies are at the far ends of the solution space w.r.t. repetition of attributes. We empirically evaluate them next in Section IV-B and discuss opportunities to improve via workload-dependent optimizations in Section V.

Horizontal Partitioning. Implicitly, so far, we have assumed that the notion of permissible leakages is defined at the level of columns and data is encrypted at that level. Such an assumption is not necessary. Consider the scenario when data values are independent/dependent is an outcome of not only the semantic meaning of the attributes but their actual values. For instance, typically, one would expect income level of a person and their education level to be correlated.

TABLE I: Preliminary experimental results. Query costs are measured in terms of the number of oblivious joins required, normalized by the number according to the Naïve baseline.

Methods	Storage	#Partitions	Query Cost
Naïve	731 MB	231	1
SNF (non-repeating)	626 MB	66	0.726
SNF (max-repeating)	14110 MB	66	0.13
Strawman	461 MB	1	0
Plaintext	30 MB	1	0

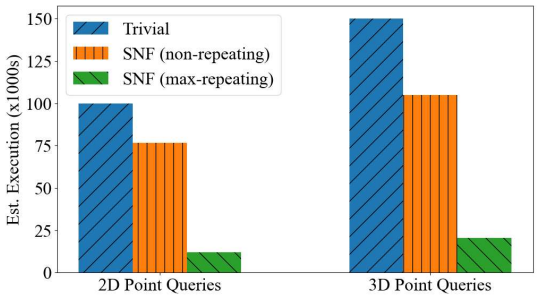

Fig. 3: Estimated query execution time cost over the joins required, comparison among different partitioning algorithms.

However, such a correlation may not hold for people in certain professions, such as a stockbroker. Likewise, data owners may have different levels of tolerance to leakages for different subsets of data belonging to the same column/attribute based on its sensitivity. For instance, leakage of the name of a disease such as cancer may be deemed significantly more sensitive than say common cold. Given the above, data owners may be willing to choose different encryption mechanisms to protect different parts of the data. Both these observations lead us to consider partitioned representation not just based on a vertical partitioning of attributes, but also horizontal partitioning. In particular, a relation may be partitioned into a subset of sub-relations which may correspond to horizontal or vertical partitioning. A partitioning would be valid as long as we can reconstruct the original relation via a sequence of join (based on the tid as discussed earlier in the context of vertical partitioning) or union operators (in case of horizontal partitioning). Such an expanded scope of partitioning significantly increases our repertoire of data representations. For instance, we may partition a table horizontally based on a data value, if it results in a given sub-relation to have a lesser number of data dependencies (e.g., two attribute values are dependent in general, but are independent based on a value of some attribute, and the relation is horizontally partitioned based on that attribute value). Now we may need to vertically partition only one of the sub-relations leaving the attributes collocated in the other horizontally partitioned subrelation. Likewise, horizontal partitioning can also be exploited to keep parts of data collocated if data's sensitivity level depends upon its value or that of other collocated attributes. In general, a data representation may correspond to a partially ordered set of horizontal and vertical partitioning and can be represented as a tree and the space of partitioning becomes that of all such trees. To handle horizontal partitioning in conjunction with vertical partitioning, we need to extend the reasoning of leakages, the definition of SNF, and also the concept of maximally permissive leakage to such representations.

B. Preliminary Experimental Results

As a proof-of-concept, we implement and evaluate different partitioning strategies in our system (as described in Algorithm 1) to pitch the feasibility and potential interests of our vision on secure normal forms.

Dataset. We present our preliminary empirical results on the 2013 U.S. Census American Community Survey (ACS) dataset [51]. The ACS dataset contains one-year survey responses of individuals of 231 attributes and 153,589 records, a larger version of this dataset being used in prior research [31] to demonstrate the effectiveness of the inference attacks.

Partitioning Algorithms. The partitioning algorithms we implement over ACE dataset are the trivial/naïve partitioning and the *non-repeating* and *max-repeating* strategies discussed in the previous section. These algorithms are compared against storing the database in plaintext and the strawman solution. In the strawman solution, we have all encrypted-as-specified columns in one single relation (simulating the naïve usage of CryptDB [6]). To do so, we randomly sample 172 attributes to encrypt with weak encryption techniques such as DET and OPE, and the rest of the attributes are encrypted with AES.

Query workload. For simplicity, we assume the queries in the workload have the following template:

```
SELECT attr_1,
FROM relations(attr_1, attr_2, attr_3,..)
WHERE predicate(attr_2, attr_3)
```

To generate this query, which we call a 2-way point query, we randomly select two columns that are weakly encrypted and project to another random attribute. Likewise, we create 3-way point queries. We choose 100 different 2-way point queries and other 100 different 3-way point queries to form the workload.

Goals. With our experiments, we would like to discuss the following:

- *Overhead of the Partitioning Strategies.* We first investigate the storage overhead, incurred by encryption and different partitions. Particularly, in this early study, we focus on showing the tradeoffs among different partitioning methods in the solution space, while comparing with strawman solutions and trivial/naïve baselines, (to inspire more future research in this field). We relatively simplify the data correlation and inference model of leakages by considering only functional dependencies as the inference channel.
- *Query Execution Cost.* The query cost model estimates the cost of executing the queries over the dataset in terms of the number of oblivious "join" operations.

Exp 1: The Cost of Partitioning Strategies. Enabling encryption over ACS dataset using the "Strawman" solution incurs

15x storage overhead compared to storing the plaintext table ("Plaintext"). As expected, the naïve normalization strategy almost doubles the storage cost than the strawman solution, because it partitions every attribute into a single sub-relation and properly adds the encrypted tid column, which results in 231 sub-relations. The two SNF strategies reduce the number of partitions greatly from 231 to 66, and the storage overhead varies between 626 MB to 14110 MB (0.85x – 19x compared to the naïve baseline), leaving a large space for optimization and future improvement.

Exp 2: Query Execution Cost. From the workload perspective, every query executed over the naïve partitioning requires a 2-way or 3-way oblivious join. In contrast, the SNF strategies (non-repeating and max-repeating) satisfy maximal permissiveness and reduce the query execution cost to only 0.726x – 0.13x compared to the naïve solution. We further plot the estimated execution time, based on existing oblivious join algorithms [52], for the join operators over the three partitioning methods (see Figure 3). The discrepancy in terms of the time cost between partitioning methods is observed to be large, which shows opportunities for workload-aware optimizations, discussed in the next section. We note that the discussion only includes very preliminary results of SNFs. Designing and implementing an optimal query execution plan with respect to a given SNF which avoids any unintended dynamic leakages is critical to the system performance and security Opportunities to further optimize the system performance when incorporating SNFs are observed and discussed in the next section.

V. Opportunities and Challenges with SNFs

The SNF vision opens the following opportunities for secure data management and outsourcing, and challenges for the full realization of a practical system.

A. Leakage Characterization

While in this paper, we do not distinguish leakages from different weak encryption techniques but focus on the spreading of leakages, the information leaked by different encryption techniques can be measured in different ways. Prior work in cryptography [53]–[55] has studied and listed a number of syntactic leakages based on the structural properties of different cryptographic algorithms (e.g., data volume, query length, response size, etc.). However, such definitions, in nature, are not compatible with database inferences based on data semantics (e.g., functional dependencies, correlations). While being difficult, we would like to develop an overarching system to connect two types of leakages altogether through a holistic modeling of leakages.

One way to achieve this is to characterize the leakages as the result of cryptography precisely in the semantics sense. We can define the leakages from the perspective of *association, relationship*, and *distribution*. An association leakage arises when the adversary is able to associate a ciphertext representation with one plaintext value more confidently than with another. Relationship leakages measure the leakage of the l-ary relationship of any subset of plaintext values from their ciphertext representations, whereas distribution leakages characterize the leakage of plaintext distributions. Other types of leakages can be developed as well. The leakage from one cryptographic technique could comprise one or more leakages from the ones defined above. Having this characterization can devise a more fine-grained leakage analysis to replace the *simplified conservative* assumption in the leakage inference module and thus improve the final SNF representation.

Quantifying Leakages. Rather than measuring a leakage as a boolean value, we can quantify it by the exact amount of information learned by the adversary. Such quantification can potentially speed up query execution or incur less number of sub-relations in SNF when secure. Our previous work [56], [57] on inference control over access control protected data, based on a plausible deniability model, establishes a threshold on how many elements of the domain could be allowed to leak. Similar ideas could be explored in SNFs if leakages from different encryption techniques could be uniformly characterized. With precisely captured partial leakages, SNF could provide the knob to users who wish to tune between security and desired functionalities/efficiency. And then, we may leverage the state-of-the-art inference attacks [27], [31] to simulate an adversary to measure the extent to which the data can be recovered.

Acquisition of Knowledge. The correctness of our system depends on the knowledge of dependence or independence. Acquiring such knowledge, as discussed earlier, could be an automatic learning process from data or human-generated rules based on schema, or a mixture of both. Note that there could be exponentially many dependencies among attributes in the schema. Missing a dependency may leak more than required. Therefore, an incomplete acquisition can affect the system's security or performance, depending on whether the system assumes full dependence or full independence by default. We would like to initiate a user interface or mechanism in the system that opts to the user to choose to be pessimistic or optimistic. It is interesting to explore the impact on how well the adversary can perform an inference attack between these two modes.

B. Partitioning and Enumeration

While our secure normalization is similar to database partitioning [32], [33], [42], [58] in spirit, there are huge opportunities in designing optimal secure normalization algorithms. First, unlike partitioning in database design, SNFs allow repetitions of data attributes to be encrypted using different cryptographic techniques. This enlarges the solution space even for vertical partitioning only. Furthermore, if we consider vertical and horizontal partitioning, the solution space becomes super-exponential. Our early experiments only empirically demonstrate the two partitioning strategies, especially limited to vertical partitioning only. While it remains exploration on a normalization algorithm to perform vertical and horizontal partitioning, the trade-off (i.e., a trade-off between 0.85x –

19x storage overhead and 0.726x – 0.13x workload execution overhead) between the two vertical strategies has shown opportunities for optimization for a given workload.

Workload-Aware Partitioning. If we know in advance a representative workload W that captures the queries to be asked on the outsourced database in the future, we can partition the database with the aid of this query workload. This is likely to happen because, in the data outsourcing scenario, the data owner is usually the same as the queried. Then the partitioning process becomes solving an optimization problem

$$\{R_1, \ldots, R_n\} = \arg\min_{P_i \in \mathcal{P}} Cost(W, P_i), \text{s.t. SNF properties}$$

where P_i is a candidate partitioning representation from all possibilities $\mathcal{P}$. We would like to find a set of sub-relations that minimize the estimated cost of executing the workload over them while satisfying the properties of no unintended leakages.

Migration to Other Workloads. However, chances are the representative workload may change over time after the database is outsourced. A natural research question is how to migrate the partitioned database in SNFs to another SNF representation optimized for the new workload. A naïve answer to this question may be re-doing the partitioning entirely and updating the outsourced database, which will inevitably incur overhead. While this leaves future exploration on algorithmic design to solve the problem, it is noteworthy that the adversary can observe the update process in which the query information could be leaked. It remains an open question to defend against such adversaries in the migration process.

SNF over Dynamic Databases. The data outsourced can change over time (i.e., the database can grow or shrink when data is imported, updated or deleted). Dynamic data update is a challenging problem for encrypted databases. The current design of SNF may require recasting and re-partitioning of the encrypted data to be securely outsourced. Updating SNF without such operations is an interesting open question for future work.

C. Secure Query Execution

Data-Aware Sub-Relation Matching. Given a query, there may be multiple collections of sub-relations in SNF to execute on. For example, a database $R(A, B, C, D)$ can be partitioned in to $R_1(A, B), R_2(B, D), R_3(A, C)$. For query $q(A, D)$, it could be executed over $R_1 \otimes R_2$ or $R_2 \otimes R_3$. Selecting one collection rather than the other may reduce some execution costs, which would be preferred. This question is related to the query-view answering problem [59] of finding a materialized view to answer a given query. However, in our context, designing proper data structures and protocols to enable an optimized matching without leakage still requires further exploration.

Towards Multi-Relational Queries. We have considered SNF when considering a relation being stored using multiple cryptographic primitives. When data is stored in SNF, adversaries cannot learn anything other than what the data owner specified as permissible leakages based on the choice of encryption they used. Furthermore, we have discussed how queries can be executed by reconstructing (parts of) a relation needed to answer the query without leaking linkability and thus ensuring that no further leakage occurs other than those permissible. A full solution, however, requires generalizing our approach to support multiple tables, each in SNF based on appropriate choice of cryptography, and moreover, to support techniques to compose relational operators over multiple encrypted relations. Our discussions have focused on preventing leakages in the context of reconstructing a single relation from its partitioned storage. We will need to generalize the approach to further explore and prevent leakages when dealing with general multi-relation queries.

Optimization and Trade-offs in Query Execution. As shown in previous work [15], [48], [49] and our early results, executing the oblivious join operators is time-costly. Independent of the oblivious processing and query binning thoughts, another optimization may be enabling some local data structures that help the server obliviously shrink the table sizes to be joined with a few rounds of communication trade-off.

D. Towards Real-World Deployment

In the future, we would like to extend our SNF vision to a fully functioning system that can be deployed in real-world use cases. We identify a few challenges toward a usable and practical system.

Visualizing Leakages. Understanding permissible leakages as the nature of cryptographic techniques and how unintended leakages can happen through data dependencies requires domain knowledge. It may be challenging for non-domain experts to explore data outsourcing or correctly specify the amount and types of leakages they can tolerate. Building a visual interface that can allow users to specify database operators and cryptographic techniques, and compare the leakages with immediate system feedback through inference would make the system more usable.

Language for Leakage on Representations. To visualize leakages, one may require a uniform language to represent leakages, given the heterogeneous nature of leakages from different cryptographic techniques, as discussed earlier. We envision such a language to bridge the gap between the leakage/cryptography specification and the symbolic inference rules for leakage reasoning.

Leakage as Indexing. From the query execution perspective, the information leaked through the cryptographic technique enforced over a column enables faster or cheaper database operations on encrypted data. That said, leakages can be viewed as "indexing" that user can explore to enforce in their outsourcing database. Exploring this functionality may be able to be implemented in our envisioned system.

Other Types of Leakages. We have so far discussed leakages due to using multiple encryption techniques on data. To build an end-to-end system, we need to deal with different types of leakages, such as leakages from metadata [60], leakages from

the timing of the operations (e.g., inserting new data) [61], leakages from database logs [62]–[64], leakage from authentication algorithms [29], [30], leakages form commitment algorithms [65], etc. Building new primitives for encrypted meta-data to mitigate such leakages would make the system more robust. How such techniques interact and compose with SNF-based representation of data when multiple cryptographic primitives are used in the same system would be an interesting direction for exploration.

Beyond Relational Databases. The modern database designs have gone beyond the relational tables. Data can be stored as documents, key-value pairs, geo-location [66]–[68] node-edge adjacency representations [69], or even vectorized embeddings [69]–[72] in the databases. While the design of SNFs is much inspired by the theoretical development of relational database (integrity) constraints, the secure representation of data should be independent of how data is stored. How to answer queries securely over SNFs with data stored in a non-relational database is still an unanswered and interesting research question for future development.

System Deployment. Once a full prototype system is implemented, deploying it in real-world use cases would be interesting. In future work, we would like to integrate our system as a middle-ware framework into Cisco's cloud database product, Panoptica [35], and test the system's performance with real-world query workloads. Panoptica supports cloud-based secure solutions and several use cases. Extensive experimentation may provide evidence of whether the system can efficiently detect leakages, execute the partitioning algorithm, and answer queries over SNFs. Thus, we can answer more research questions, such as which cryptographic techniques can be used under different leakages and database operators (e.g., SPJA queries), given constraints on available system resources.

E. Relationship to Differential Privacy.

The connection between SNF and DP for databases [73] can be seen in two dimensions. Firstly, prior work [74]–[76] investigates weak encryption techniques with a differentially private leakage, which can be easily quantified and composed. Hence, we can build our SNFs on top of similar cryptographic techniques with differential privacy guarantees. Secondly, existing effort in building a DP query processing system [77], [78] has explored a differentially private representation of data (i.e., a DP synopsis) so that queries can be processed without consuming additional privacy budgets (i.e., post-processing). Such a representation can leverage the correlations among attributes [79], [80] or representative query workloads [77], [81], which has a connection with our SNF. Unifying these two lines of thought can be a charming direction for building a database system.

VI. Concluding Remarks: Looking Back and Looking Ahead

Looking back on the past two decades of research on encrypted databases, we believe the moment has arrived for our community to embark on a new phase of integrating piecemeal solutions into end-to-end systems. As additional, and severe, leakages can happen in this integration process, we call for special attention to this important topic, especially approaches to systematically controlling leakages whilst not much affecting system efficiency or functionalities, in this new chapter of research. This paper describes our vision and early research on understanding leakages when outsourcing encrypted databases with different cryptographic techniques. We propose *secure normal form*, a principled approach that prevents unintended leakages through inference and restricts leakage to only what the user explicitly specifies. A new system is architected to support our secure normal form vision, with a modular design of replacing several parts of the current database, including partitioning, outsourcing, and query answering. Our early empirical results show the potential interests and usefulness of secure normal forms. Opportunities and challenges are discussed toward fully realizing this system so that it can be practically deployed in the current secure cloud outsourcing industry.

Acknowledgment

We would like to thank the anonymous reviewers for their constructive and detailed feedback. The work of Xi He is supported by NSERC through a Discovery Grant. The work of Sharad Mehrotra is supported by NSF grants 2420846, 2245372, 2133391, 2008993, and 1952247. The work of Shantanu Sharma is supported by NSF grant 2245374. The work of Ashish Kundu is supported by Cisco Inc. We also thank Cisco Research for the research grant award.

References

[1] H. Hacigümüs, B. R. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002*, M. J. Franklin, B. Moon, and A. Ailamaki, Eds. ACM, 2002, pp. 216–227. [Online]. Available: https://doi.org/10.1145/564691.564717

[2] D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000*. IEEE Computer Society, 2000, pp. 44–55. [Online]. Available: https://doi.org/10.1109/SECPRI.2000.848445

[3] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart, "Deterministic encryption: Definitional equivalences and constructions without random oracles," in *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, ser. Lecture Notes in Computer Science, D. A. Wagner, Ed., vol. 5157. Springer, 2008, pp. 360–378. [Online]. Available: https://doi.org/10.1007/978-3-540-85174-5_20

[4] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, ser. Lecture Notes in Computer Science, A. Menezes, Ed., vol. 4622. Springer, 2007, pp. 535–552. [Online]. Available: https://doi.org/10.1007/978-3-540-74143-5_30

[5] A. Boldyreva, S. Fehr, and A. O'Neill, "On notions of security for deterministic encryption, and efficient constructions without random oracles," in *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, ser. Lecture Notes in Computer Science, D. A. Wagner, Ed., vol. 5157. Springer, 2008, pp. 335–359. [Online]. Available: https://doi.org/10.1007/978-3-540-85174-5_19

[6] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in *Proceedings of the 23rd ACM Symposium on Operating Systems Principles 2011, SOSP 2011, Cascais, Portugal, October 23-26, 2011*, T. Wobber and P. Druschel, Eds. ACM, 2011, pp. 85–100. [Online]. Available: https://doi.org/10.1145/2043556.2043566

[7] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, 2011, pp. 578–595. [Online]. Available: https://doi.org/10.1007/978-3-642-22792-9_33

[8] F. Kerschbaum, "Frequency-hiding order-preserving encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, I. Ray, N. Li, and C. Kruegel, Eds. ACM, 2015, pp. 656–667. [Online]. Available: https://doi.org/10.1145/2810103.2813629

[9] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[10] ——, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, M. Mitzenmacher, Ed. ACM, 2009, pp. 169–178. [Online]. Available: https://doi.org/10.1145/1536414.1536440

[11] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, ser. Lecture Notes in Computer Science, T. Rabin, Ed., vol. 6223. Springer, 2010, pp. 465–482. [Online]. Available: https://doi.org/10.1007/978-3-642-14623-7_25

[12] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, vol. 53, no. 3, pp. 97–105, 2010. [Online]. Available: https://doi.org/10.1145/1666420.1666444

[13] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu, "Private database queries using somewhat homomorphic encryption," in *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, ser. Lecture Notes in Computer Science, M. J. J. Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, Eds., vol. 7954. Springer, 2013, pp. 102–118. [Online]. Available: https://doi.org/10.1007/978-3-642-38980-1_7

[14] M. Blum and S. Goldwasser, "An efficient probabilistic public-key encryption scheme which hides all partial information," in *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds., vol. 196. Springer, 1984, pp. 289–302. [Online]. Available: https://doi.org/10.1007/3-540-39568-7_23

[15] Y. Wang and K. Yi, "Secure yannakakis: Join-aggregate queries over private data," in *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, G. Li, Z. Li, S. Idreos, and D. Srivastava, Eds. ACM, 2021, pp. 1969–1981. [Online]. Available: https://doi.org/10.1145/3448016.3452808

[16] Q. Luo, Y. Wang, K. Yi, S. Wang, and F. Li, "Secure sampling for approximate multi-party query processing," *Proc. ACM Manag. Data*, vol. 1, no. 3, pp. 219:1–219:27, 2023. [Online]. Available: https://doi.org/10.1145/3617339

[17] Y. Sun, S. Wang, H. Li, and F. Li, "Building enclave-native storage engines for practical encrypted databases," *Proc. VLDB Endow.*, vol. 14, no. 6, pp. 1019–1032, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p1019-sun.pdf

[18] B. Fisch, D. Vinayagamurthy, D. Boneh, and S. Gorbunov, "IRON: functional encryption using intel SGX," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 765–782. [Online]. Available: https://doi.org/10.1145/3133956.3134106

[19] A. Papadimitriou, R. Bhagwan, N. Chandran, R. Ramjee, A. Haeberlen, H. Singh, A. Modi, and S. Badrinarayanan, "Big data analytics over encrypted datasets with seabed," in *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, 2016, pp. 587–602. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/papadimitriou

[20] S. Bian, Z. Zhang, H. Pan, R. Mao, Z. Zhao, Y. Jin, and Z. Guan, "HE3DB: an efficient and elastic encrypted database via arithmetic-and-logic fully homomorphic encryption," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 2930–2944. [Online]. Available: https://doi.org/10.1145/3576915.3616608

[21] Z. He, W. K. Wong, B. Kao, D. W. Cheung, R. Li, S. Yiu, and E. Lo, "SDB: A secure query processing system with data interoperability," *Proc. VLDB Endow.*, vol. 8, no. 12, pp. 1876–1879, 2015. [Online]. Available: http://www.vldb.org/pvldb/vol8/p1876-he.pdf

[22] D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright, "From keys to databases - real-world applications of secure multi-party computation," *Comput. J.*, vol. 61, no. 12, pp. 1749–1771, 2018. [Online]. Available: https://doi.org/10.1093/comjnl/bxy090

[23] P. Antonopoulos, A. Arasu, K. D. Singh, K. Eguro, N. Gupta, R. Jain, R. Kaushik, H. Kodavalla, D. Kossmann, N. Ogg, R. Ramamurthy, J. Szymaszek, J. Trimmer, K. Vaswani, R. Venkatesan, and M. Zwilling, "Azure SQL database always encrypted," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, D. Maier, R. Pottinger, A. Doan, W. Tan, A. Alawini, and H. Q. Ngo, Eds. ACM, 2020, pp. 1511–1525. [Online]. Available: https://doi.org/10.1145/3318464.3386141

[24] Amazon Inc., "AWS clean rooms documentation," https://docs.aws.amazon.com/clean-rooms/, accessed: 2024-02-28.

[25] MongoDB Inc., "MongoDB queryable encryption documentation," https://www.mongodb.com/docs/manual/core/queryable-encryption/, accessed: 2024-02-28.

[26] R. A. Popa, N. Zeldovich, and H. Balakrishnan, "Guidelines for using the cryptdb system securely," *IACR Cryptol. ePrint Arch.*, p. 979, 2015. [Online]. Available: http://eprint.iacr.org/2015/979

[27] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, I. Ray, N. Li, and C. Kruegel, Eds. ACM, 2015, pp. 644–655. [Online]. Available: https://doi.org/10.1145/2810103.2813651

[28] Z. Gui, K. G. Paterson, and T. Tang, "Security analysis of mongodb queryable encryption," in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, J. A. Calandrino and C. Troncoso, Eds. USENIX Association, 2023, pp. 7445–7462. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/gui

[29] A. Kundu, M. J. Atallah, and E. Bertino, "Leakage-free redactable signatures," in *Second ACM Conference on Data and Application Security and Privacy, CODASPY 2012, San Antonio, TX, USA, February 7-9, 2012*, E. Bertino and R. S. Sandhu, Eds. ACM, 2012, pp. 307–316. [Online]. Available: https://doi.org/10.1145/2133601.2133639

[30] A. Kundu and E. Bertino, "Structural signatures for tree data structures," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 138–150, 2008. [Online]. Available: http://www.vldb.org/pvldb/vol1/1453876.pdf

[31] V. Bindschaedler, P. Grubbs, D. Cash, T. Ristenpart, and V. Shmatikov, "The tao of inference in privacy-protected databases," *Proc. VLDB Endow.*, vol. 11, no. 11, pp. 1715–1728, 2018. [Online]. Available: http://www.vldb.org/pvldb/vol11/p1715-bindschaedler.pdf

[32] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu, "Two can keep A secret: A distributed architecture for secure database services," in *Second Biennial Conference on Innovative Data Systems Research, CIDR 2005, Asilomar, CA, USA, January 4-7, 2005, Online Proceedings*. www.cidrdb.org, 2005, pp. 186–199. [Online]. Available: http://cidrdb.org/cidr2005/papers/P16.pdf

[33] A. Deshpande, "Sypse: Privacy-first data management through pseudonymization and partitioning," in *11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings*. www.cidrdb.org, 2021. [Online]. Available: http://cidrdb.org/cidr2021/papers/cidr2021_paper24.pdf

[34] M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems, Third Edition*. Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-1-4419-8834-8

[35] "Panoptica: Cisco cloud application security," https://www.panoptica.app/, accessed: 2023-11-22.

[36] S. Patel, G. Persiano, K. Yeo, and M. Yung, "Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM, 2019, pp. 79–93. [Online]. Available: https://doi.org/10.1145/3319535.3354213

[37] M. Studeny and R. R. Bouckaert, "On chain graph models for description of conditional independence structures," *Annals of Statistics*, pp. 1434–1495, 1998.

[38] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning bayesian networks: The combination of knowledge and statistical data," *Mach. Learn.*, vol. 20, no. 3, pp. 197–243, 1995. [Online]. Available: https://doi.org/10.1007/BF00994016

[39] D. R. Karger and N. Srebro, "Learning markov networks: maximum bounded tree-width graphs," in *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, S. R. Kosaraju, Ed. ACM/SIAM, 2001, pp. 392–401. [Online]. Available: http://dl.acm.org/citation.cfm?id=365411.365486

[40] A. R. Klivans and R. Meka, "Learning graphical models using multiplicative weights," in *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, C. Umans, Ed. IEEE Computer Society, 2017, pp. 343–354. [Online]. Available: https://doi.org/10.1109/FOCS.2017.39

[41] D. Koller and N. Friedman, *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009. [Online]. Available: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11886

[42] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Addison-Wesley, 1995. [Online]. Available: http://webdam.inria.fr/Alice/

[43] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012. [Online]. Available: https://www.ndss-symposium.org/ndss2012/access-pattern-disclosure-searchable-encryption-ramification-attack-and-mitigation

[44] S. Han, V. Chakraborty, M. T. Goodrich, S. Mehrotra, and S. Sharma, "Veil: A storage and communication efficient volume-hiding algorithm," *Proc. ACM Manag. Data*, vol. 1, no. 4, pp. 265:1–265:27, 2023. [Online]. Available: https://doi.org/10.1145/3626759

[45] S. Maiyya, S. Ibrahim, C. Scarberry, D. Agrawal, A. E. Abbadi, H. Lin, S. Tessaro, and V. Zakhary, "Quoram: A quorum-replicated fault tolerant ORAM datastore," in *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, K. R. B. Butler and K. Thomas, Eds. USENIX Association, 2022, pp. 3665–3682. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/maiyya

[46] E. Stefanov, M. van Dijk, E. Shi, C. W. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path ORAM: an extremely simple oblivious RAM protocol," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, A. Sadeghi, V. D. Gligor, and M. Yung, Eds. ACM, 2013, pp. 299–310. [Online]. Available: https://doi.org/10.1145/2508859.2516660

[47] W. Zheng, A. Dave, J. G. Beekman, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Opaque: An oblivious and encrypted distributed analytics platform," in *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, A. Akella and J. Howell, Eds. USENIX Association, 2017, pp. 283–298. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zheng

[48] H. Pang and X. Ding, "Privacy-preserving ad-hoc equi-join on outsourced data," *ACM Trans. Database Syst.*, vol. 39, no. 3, pp. 23:1–23:40, 2014. [Online]. Available: https://doi.org/10.1145/2629501

[49] Z. Chang, D. Xie, S. Wang, and F. Li, "Towards practical oblivious join," in *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Z. G. Ives, A. Bonifati, and A. E. Abbadi, Eds. ACM, 2022, pp. 803–817. [Online]. Available: https://doi.org/10.1145/3514221.3517868

[50] S. Mehrotra, S. Sharma, J. D. Ullman, D. Ghosh, P. Gupta, and A. Mishra, "PANDA: partitioned data security on outsourced sensitive and non-sensitive data," *ACM Trans. Manag. Inf. Syst.*, vol. 11, no. 4, pp. 23:1–23:41, 2020. [Online]. Available: https://doi.org/10.1145/3397521

[51] U. C. Bureau, "American community survey (ACS)," Last Accessed: 2023-09-07. [Online]. Available: https://www.census.gov/programs-surveys/acs/data.html

[52] Y. Wang and K. Yi, "Secure yannakakis: Join-aggregate queries over private data," in *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, G. Li, Z. Li, S. Idreos, and D. Srivastava, Eds. ACM, 2021, pp. 1969–1981. [Online]. Available: https://doi.org/10.1145/3448016.3452808

[53] S. Kamara, T. Moataz, and O. Ohrimenko, "Structured encryption and leakage suppression," in *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, ser. Lecture Notes in Computer Science, H. Shacham and A. Boldyreva, Eds., vol. 10991. Springer, 2018, pp. 339–370. [Online]. Available: https://doi.org/10.1007/978-3-319-96884-1_12

[54] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, ser. Lecture Notes in Computer Science, M. Abe, Ed., vol. 6477. Springer, 2010, pp. 577–594. [Online]. Available: https://doi.org/10.1007/978-3-642-17373-8_33

[55] M. George, S. Kamara, and T. Moataz, "Structured encryption and dynamic leakage suppression," in *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III*, ser. Lecture Notes in Computer Science, A. Canteaut and F. Standaert, Eds., vol. 12698. Springer, 2021, pp. 370–396. [Online]. Available: https://doi.org/10.1007/978-3-030-77883-5_13

[56] P. Pappachan, S. Zhang, X. He, and S. Mehrotra, "Don't be a tattle-tale: Preventing leakages through data dependencies on access control protected data," *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 2437–2449, 2022. [Online]. Available: https://www.vldb.org/pvldb/vol15/p2437-pappachan.pdf

[57] ——, "Preventing inferences through data dependencies on sensitive data," *IEEE Transactions on Knowledge and Data Engineering*, to appear. [Online]. Available: https://doi.org/10.1109/TKDE.2023.3336630

[58] V. Ganapathy, D. Thomas, T. Feder, H. Garcia-Molina, and R. Motwani, "Distributing data for secure database services," *Trans. Data Priv.*, vol. 5, no. 1, pp. 253–272, 2012. [Online]. Available: http://www.tdp.cat/issues11/abs.a089a11.php

[59] A. Y. Halevy, "Answering queries using views: A survey," *VLDB J.*, vol. 10, no. 4, pp. 270–294, 2001. [Online]. Available: https://doi.org/10.1007/s007780100054

[60] W. Chen and R. A. Popa, "Metal: A metadata-hiding file-sharing system," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/metal-a-metadata-hiding-file-sharing-system/

[61] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," *IEEE Transactions on knowledge and data engineering*, vol. 23, no. 1, pp. 51–63, 2010.

[62] S. Shastri, V. Banakar, M. Wasserman, A. Kumar, and V. Chidambaram, "Understanding and benchmarking the impact of GDPR on database systems," *Proc. VLDB Endow.*, vol. 13, no. 7, pp. 1064–1077, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p1064-shastri.pdf

[63] V. Chakraborty, S. Ann-Elvy, S. Mehrotra, F. Nawab, M. Sadoghi, S. Sharma, N. Venkatasubramanian, and F. Saeed, "Data-case: Grounding data regulations for compliant data processing systems," in *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, L. Tanca, Q. Luo, G. Polese, L. Caruccio, X. Oriol, and D. Firmani, Eds. OpenProceedings.org, 2024, pp. 108–115. [Online]. Available: https://doi.org/10.48786/edbt.2024.10

[64] P. Pappachan, R. Yus, S. Mehrotra, and J. Freytag, "Sieve: A middleware approach to scalable access control for database management systems," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2424–2437, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p2424-pappachan.pdf

[65] X. Li, C. Weng, Y. Xu, X. Wang, and J. Rogers, "ZKSQL: verifiable and efficient query evaluation with zero-knowledge proofs," *Proc. VLDB*

Endow., vol. 16, no. 8, pp. 1804–1816, 2023. [Online]. Available: https://www.vldb.org/pvldb/vol16/p1804-li.pdf

[66] L. Yu, S. Zhang, L. Zhou, Y. Meng, S. Du, and H. Zhu, "Thwarting longitudinal location exposure attacks in advertising ecosystem via edge computing," in *42nd IEEE International Conference on Distributed Computing Systems, ICDCS 2022, Bologna, Italy, July 10-13, 2022*. IEEE, 2022, pp. 470–480. [Online]. Available: https://doi.org/10.1109/ICDCS54860.2022.00052

[67] L. Yu, S. Zhang, Y. Meng, S. Du, Y. Chen, Y. Ren, and H. Zhu, "Privacy-preserving location-based advertising via longitudinal geo-indistinguishability," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2023.

[68] P. Pappachan, V. S. H. Manjunath, C. Qiu, A. C. Squicciarini, and H. Onweller, "CORGI: an interactive framework for customizable and robust location obfuscation," in *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 2023, pp. 3667–3670. [Online]. Available: https://doi.org/10.1109/ICDE55515.2023.00294

[69] G. Jin, X. Feng, Z. Chen, C. Liu, and S. Salihoglu, "Kùzu graph database management system," in *13th Conference on Innovative Data Systems Research, CIDR 2023, Amsterdam, The Netherlands, January 8-11, 2023*. www.cidrdb.org, 2023. [Online]. Available: https://www.cidrdb.org/cidr2023/papers/p48-jin.pdf

[70] Z. Hu, X. Li, D. P. Woodruff, H. Zhang, and S. Zhang, "Recovery from non-decomposable distance oracles," *IEEE Transactions on Information Theory*, vol. 69, no. 10, pp. 6443–6469, 2023.

[71] S. Han, Q. Zhang, Y. Yao, W. Jin, Z. Xu, and C. He, "LLM multi-agent systems: Challenges and open problems," *CoRR*, vol. abs/2402.03578, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.03578

[72] D. Kang, J. Guibas, P. Bailis, T. Hashimoto, Y. Sun, and M. Zaharia, "Data management for ml-based analytics and beyond," *ACM / IMS J. Data Sci.*, vol. 1, no. 1, jan 2024. [Online]. Available: https://doi.org/10.1145/3611093

[73] J. P. Near and X. He, "Differential privacy for databases," *Found. Trends Databases*, vol. 11, no. 2, pp. 109–225, 2021. [Online]. Available: https://doi.org/10.1561/1900000066

[74] X. Li, Y. Hu, W. Liu, H. Feng, L. Peng, Y. Hong, K. Ren, and Z. Qin, "Opboost: A vertical federated tree boosting framework based on order-preserving desensitization," *Proc. VLDB Endow.*, vol. 16, no. 2, pp. 202–215, 2022. [Online]. Available: https://www.vldb.org/pvldb/vol16/p202-li.pdf

[75] A. R. Chowdhury, B. Ding, S. Jha, W. Liu, and J. Zhou, "Strengthening order preserving encryption with differential privacy," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 2519–2533. [Online]. Available: https://doi.org/10.1145/3548606.3560610

[76] Y. Zhang, J. Bater, K. Nayak, and A. Machanavajjhala, "Longshot: Indexing growing databases using MPC and differential privacy," *Proc. VLDB Endow.*, vol. 16, no. 8, pp. 2005–2018, 2023. [Online]. Available: https://www.vldb.org/pvldb/vol16/p2005-zhang.pdf

[77] I. Kotsogiannis, Y. Tao, X. He, M. Fanaeepour, A. Machanavajjhala, M. Hay, and G. Miklau, "Privatesql: A differentially private SQL query engine," *Proc. VLDB Endow.*, vol. 12, no. 11, pp. 1371–1384, 2019. [Online]. Available: http://www.vldb.org/pvldb/vol12/p1371-kotsogiannis.pdf

[78] S. Zhang and X. He, "Dprovdb: Differentially private query processing with multi-analyst provenance," *Proc. ACM Manag. Data*, vol. 1, no. 4, pp. 267:1–267:27, 2023. [Online]. Available: https://doi.org/10.1145/3626761

[79] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Privbayes: private data release via bayesian networks," in *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, C. E. Dyreson, F. Li, and M. T. Özsu, Eds. ACM, 2014, pp. 1423–1434. [Online]. Available: https://doi.org/10.1145/2588555.2588573

[80] E. Bao, X. Xiao, J. Zhao, D. Zhang, and B. Ding, "Synthetic data generation with differential privacy via bayesian networks," *J. Priv. Confidentiality*, vol. 11, no. 3, 2021. [Online]. Available: https://doi.org/10.29012/jpc.776

[81] Y. Xiao, G. Wang, D. Zhang, and D. Kifer, "Answering private linear queries adaptively using the common mechanism," *Proc. VLDB Endow.*, vol. 16, no. 8, pp. 1883–1896, 2023. [Online]. Available: https://www.vldb.org/pvldb/vol16/p1883-xiao.pdf