Enforcing Privacy in Cloud Databases

Somayeh Sobati Moghadam, Jérôme Darmont, Gérald Gavin

Enforcing Privacy in Cloud Databases

Somayeh Sobati Moghadam, Jérôme Darmont, and Gérald Gavin

Université de Lyon, Lyon 2, Lyon 1, ERIC EA3083
5 avenue Pierre Mendès France – 69676 Bron Cedex – France
ssobati@eric.univ-lyon2.fr, jerome.darmont@univ-lyon2.fr, gerald.gavin@univ-lyon1.fr

Abstract. Outsourcing databases, i.e., resorting to Database-as-a-Service (DBaaS), is nowadays a popular choice due to the elasticity, availability, scalability and pay-as-you-go features of cloud computing. However, most data are sensitive to some extent, and data privacy remains one of the top concerns to DBaaS users, for obvious legal and competitive reasons. In this paper, we survey the mechanisms that aim at making databases secure in a cloud environment, and discuss current pitfalls and related research challenges.

Keywords: Databases, Cloud Computing, DBaaS, Data Privacy, Data Encryption

1 Introduction

Cloud computing offers a variety of services via a pay-per-use model on the Internet. The flexibility that cloud computing offers is very appealing for many organizations, especially mid-sized and small ones, because it provides reduced start-up costs and means to financially cope with variations in system usage. Outsourcing data to the cloud is particularly interesting [75]. However, some data are especially sensitive, e.g., personal data, health-related data, business data, and generally data used in decision-support processes. Outsourcing a database in the cloud raises security issues, some related to cloud architectures (e.g., untrusted service providers, curious cloud employees...), and others related to such concerns as data privacy, integrity and availability. With increasingly sophisticated internal and external cloud attacks, traditional security mechanisms are no longer sufficient to protect cloud databases [77].

Let us consider a Database-as-a-Service (DBaaS) scenario (Figure 1) where a user outsources a database at one or more Cloud Service Providers' (CSPs). The objective is to eliminate storage and minimize computation at the user's to take full advantage of cloud benefits. Yet, anything beyond the user is considered untrusted. CSPs might indeed be honest but curious, i.e., read the user's data, or even be malicious or maliciously hacked, i.e., alter data or provide fake query results. Network transactions are also considered unsafe. The user must thus protect sensitive data and queries before sending them to CSPs, and safely reconstruct query results "at home".

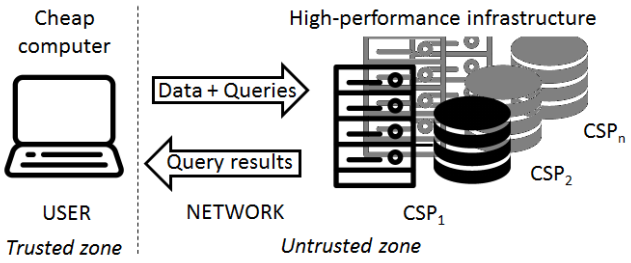

Fig. 1. Database outsourcing scenario

In this paper, we survey the security mechanisms that may be exploited in our cloud database scenario, particularly in terms of privacy. Another recent survey intersects ours [58], but more deeply focuses on cryptography, while we adopt a broader scope on security, put more emphasis on querying efficiency and

survey non-cryptographic methods (Section 2). We also mostly target database practitioners and researchers with no background in cryptography. Moreover, we review cryptographic methods that are not covered in [58], i.e., secret sharing, Private Information Retrieval (PIR) and oblivious RAM (ORAM) schemes. We classify cryptographic tools that can be exploited within cloud database scenarios into secret sharing schemes (Section 3), index-based methods (Section 4) and secure databases (Section 5). Finally, we conclude this paper by a global discussion (Section 6).

2 Non-Cryptographic Methods

2.1 Differential Privacy

Differential privacy aims at protecting data privacy when performing statistical queries [24]. While global statistics are public, individual data must remain private. To achieve this goal, a noise term is added to statistical query results, e.g., to an average salary, thus preventing the computation of individual salaries. It would indeed be easy to compute a new salary that has just been added in the database, knowing averages avg and the number of records n in the dataset: $avg_{n+1} \times (n+1) - avg_n \times (n)$.

A randomized algorithm A enforces ϵ-differential privacy if and only if, for any two databases DB_1 and DB_2 that differ on exactly one record, the ratio between the probability that A outputs O on DB_1 and DB_2 is bounded by a constant: $P(A(DB_1) = O) \div P(A(DB_2) = O) \leq e^{\epsilon}$ [76]. The tradeoff is that the smaller ϵ is, the better privacy is enforced, but the worse accuracy is. Thus, one major challenge is reducing the amount of noise while still satisfying differential privacy. Yet, in a DBaaS context, a curious CSP would still have access to fine-grained data, which is incompatible with our data outsourcing scenario, where we aim at protecting data event from the CSP.

2.2 Data Anonymization

Data anonymization irreversibly modifies data in a way that prevents the identification of sensitive information, while allowing querying data for releasing useful statistical information [25]. Some database management systems (DBMSs) natively include anonymization schemes. For example, Oracle Data Masking Pack provides data masking for various types of data, replacing real data with realistic-looking values [51]. Obscuring query processing and results may also be achieved either by rejecting queries leading to privacy disclosure or through methods such as k-anonymity and its variants t-closeness and l-diversity, which transform k distinguishable records into k indistinguishable records [67].

In a DBaaS context, privacy-preserving queries in a distributed environment can be achieved by table perturbation and reconstruction [5]. Perturbation randomly replaces an element in a table by another with probability p. Then, reconstruction can estimate `COUNT` queries over the perturbed table. Unfortunately, other aggregation functions are not supported. Similarly, random data distortion techniques may be used, e.g., the zero-sum method provides an accurate estimation of summation for range queries [66], but it induces a trade-off between privacy and accuracy. Finally, adding different amounts of noise to query answers can be used, e.g., with iReduct [74]. iReduct initially estimates query answers and iteratively refines its estimates to minimize relative errors.

Data anonymization is less complex than encryption and straightforwardly allows querying anonymized data. However, it typically reduces data granularity, which may cause a loss of effectiveness and correctness in computation [1]. Moreover, the various anonymization methods have different impacts on data utility. As a result, the same anonymized table may provide accurate answers to some queries and inaccurate results to others. More importantly, data anonymization cannot provide an adequate level of security since private personal data can be re-identified by an adversary who has some knowledge about data [23].

2.3 Data Fragmentation

In data fragmentation [2], data are assumed not to be sensitive *per se*. What is sensitive is their association with other data. Privacy is thus guaranteed by concealing such associations with respect to a predefined set of security constraints that express restrictions on one or more attributes in a table [20, 37]. For instance, given a $Patient$ table, constraint $C = \{Name, Illness\}$ indicates that associations between patient names

and illnesses should not be disclosed. Then, the table is split into fragments such that attributes listed in a constraint belong to different fragments. For instance, table *Patient* would be split in two fragments $Patient_1$ and $Patient_2$, with $Name \in Patient_1$ and $Illness \in Patient_2$. Yet, most data fragmentation approaches apply to numerical data and specific methods most be used to handle categorical data [56].

In a cloud context, data can be partitioned at independent CSPs' with respect to security constraints. When a query is issued, an appropriate subquery is transmitted to each CSP, then the results are pieced together at the user's. Moreover, intrinsically sensitive attributes such as social security numbers are stored locally at the user's. Eventually, fragmentation-based approaches yield little overhead on query computation, but are vulnerable in cases of CSP collusion and CSP inference on data updates [35]. Additionally, retaining sensitive data at the user's requires local storage capacities, which is incompatible with our data outsourcing scenario.

3 Secret Sharing-Based Methods

Secret sharing is a particular cryptographic method introduced by Shamir in which a secret piece of data is mathematically divided into so-called shares that are stored at n participants' [60]. One single participant has no means to reconstruct the secret. A subset of $k \leq n$ participants is indeed required to reconstruct the secret, providing perfect theoretical privacy when at most $k-1$ participants collude, i.e., exchange shares. Moreover, computations can run directly on shares, outputting unintelligible results that can only be put together through k participants. Finally, up to $n-k$ participants may disappear without compromising data availability; and message authentication code or signature can be applied to guarantee data integrity. Thus, secret sharing is a promising solution to security challenges in cloud data outsourcing [35] since one can easily imagine participants being CSPs. In the following subsections, we review the secret sharing-based approaches that aim at outsourcing databases, and then discuss them globally.

3.1 Verifiable Secret Sharing

Thompson et al.'s scheme allows participants to collaboratively compute aggregation queries without gaining knowledge of intermediate results [68]. A lightweight cryptographic scheme is introduced for privacy-preserving computation and verification of `SUM` and `AVG` aggregation queries. Moreover, users can verify query results with the help of signatures, while the data values contributing to the results are kept secret from both users and the CSPs. The query issuer indeed interacts with a single CSP to obtain aggregation results and can verify whether the CSP returns correct results. However, aggregation queries other than `SUM` and `AVG` cannot be computed with this scheme.

Attasena et al. specifically target cloud data warehouses through a flexible, verifiable secret sharing scheme named fVSS [11]. fVSS minimizes shared data volume, provides inner and outer data verification to check data correctness and the honesty of CSPs, improves the ability to update shares in case of CSP failure, and adjusts share volume with respect to CSP pricing policies. Moreover, in addition to queries explicitly handled by previous schemes, fVSS also allow grouping queries that are ubiquitous in On-Line Analysis Processing (OLAP). However, although some queries can be computed directly over shares (e.g., exact match queries), others require that some or all data are decrypted first (e.g., range queries).

Wang et al. propose a framework for secure and efficient query processing of relational data in the cloud that allows exact match and range queries, as well as updates [71]. B+-tree indexes are also built to optimize query response. Both data and indexes are organized into matrices, encrypted and stored at CSPs'. Additionally, data integrity is achieved by using checksum and an index structure. This framework is robust against statistical attacks.

Statistical attacks refer to an adversary obtaining some information about ciphertexts, i.e., encrypted data, through prior knowledge about plaintexts, i.e., clear data [35]. For instance, the adversary may know plaintext distribution or frequency. Then, by extracting statistics from ciphertexts, the adversary can infer ranges containing dense data or highlight ciphertexts bearing the same frequency as plaintexts.

3.2 Order-preserving Secret Sharing

Agrawal et al. propose a complete approach to execute exact match, range, and aggregation queries over shares outsourced at multiple CSPs [3]. Original data are divided using order-preserving polynomials such

that the order of shares is the same as that of original data. However, while this solution allows efficiently processing any kind of queries, including updates, it is susceptible to statistical attacks [35].

Hadavi et al. introduce a framework to provide data privacy based on threshold secret sharing [36]. First, secret values are encrypted by an Order Preserving Encryption (OPE) scheme (Section 5.1). Then, a B+-tree is built over ciphertexts and sent to an index server. The user receives query responses, including record numbers, from the index server and can then request these records from the CSPs. As Agrawal et al.'s scheme, this approach supports different kind of queries over shares, including exact match, range and aggregation queries, as well as updates. Moreover, it provides stronger security than Agrawal et al.'s scheme. It is indeed secure against frequency attacks, and an extension uses distribution perturbation to improve its robustness against statistical attacks in general [35]; but at the price of a higher computational overhead.

3.3 Discussion

Table 1 provides a comparison of secret sharing-based database outsourcing methods with respect to the queries that can run directly on shares and security features beyond privacy and availability, i.e., integrity checks and robustness against statistical attacks.

Table 1. Comparison of secret sharing-based methods

	Allowed queries				Additional security features	
	Exact match	Range	Aggregate	Update	Integrity	Statistical attacks
Thompson et al. [68]	No	No	SUM/AVG	No	Signature	Not robust
Attasena et al. [11]	Yes	Yes	Yes	Yes	Signature	Not robust
Wang et al. [71]	Yes	Yes	Yes	Yes	Checksum	Robust
Agrawal et al. [3]	Yes	Yes	Yes	Yes	None	Not robust
Hadavi et al. [36]	Yes	Yes	Yes	Yes	None	Robust

Despite secret sharing's benefits, it is not trivial to process some queries directly over shares, especially queries requiring data ordering. Obviously, it is not efficient to send back all shares in response to a query, and execute the query over reconstructed values at the user's. However, the techniques that allow directly processing such queries as range queries over shares reveal some information about plaintexts, e.g., duplicates [3]. Moreover, since every secret is shared n times, global share size can be quite large. Communication between the user and multiple CSPs is not optimal in terms of bandwidth resources either. As a result, storage and communication overhead of secret sharing-based approaches is remarkably high for moderately large databases [44].

4 Index-Based Methods

In databases, data encryption is usually managed at the tuple level [22], which prevents any computation over ciphertexts. Thus, indexes based on plaintexts are stored together with the encrypted database to help return ciphertexts in response to queries. We distinguish three types of index-based methods, namely bucketization, order preserving indexing and indexes used in Searchable Encryption (SE) schemes. We review them in the following subsections before providing a global discussion.

4.1 Bucketization-Based Indexing

Bucketization-based indexing involves dividing data into buckets and providing explicit labels for each bucket [49]. The domain of an attribute is partitioned into a set of non-overlapping buckets. Labels may preserve the order of values in the original domain or not. They are stored along with encrypted tuples. Such indexing allows exact match, range (if data order is preserved) and join queries, but also induce false positives in query answers. Thus, query post-processing is needed at the user's to filter out false positives [59].

Hacigümüs et al. partition data as in histogram construction, e.g., by equi-depth and equi-width partitioning [33]. Then, it assigns a random tag to each bucket. Any table $T(A_1, A_2, ..., A_n)$ from a database

is stored at the CSP's as $T^S(etuple,\ A_1^S, A_2^S, ..., A_n^S)$, where *etuple* is the encrypted tuple and each A_i^S is the index of attribute A_i. Each query is rewritten into server-side and user-side subqueries Q^S and Q^C, respectively. Q^S is executed by the CSP over ciphertexts using indexes A_i^S. The result of Q^S is then sent back to the user, who decrypts it and executes Q^C to filter out false positives. Query rewriting requires maintaining metadata, including bucket labels.

With the help of an homomorphic function, this approach is extended to support aggregation queries over ciphertexts [34]. The homomorphic encryption function is based on the Privacy Homomorphism (PH) scheme [57], which relies on the difficulty of factoring large composite integers, just like the famous Rivest-Shamir-Adleman (RSA) public-key cryptosystem. Unfortunately, Mykletun and Tsudik demonstrate that the CSP can obtain plaintexts with access to ciphertexts only [49].

Based on their rebuttal of Hacigümüs et al., Mykletun and Tsudik propose a simple alternative for supporting aggregation queries [49]. The user precomputes aggregation values (e.g., `SUM` and `COUNT`) for each bucket, encrypts and stores them at the CSP's. Moreover, instead of using the PH scheme, Mykletun and Tsudik use provably secure additive homomorphic encryption schemes such as Paillier's [53] and El Gamal's [27]. Precomputing aggregations decreases security risks, but requires extra storage and makes updates more complex. Updates must indeed be executed in two steps: 1) actual data update and 2) update of related precomputed aggregates in a bucket.

Hore et al. also address shortcomings of Hacigümüs et al.'s method. They notably optimize the accuracy of range queries to minimize false positives in query results [40]. They also introduce a re-bucketization technique, in which the user can fine-tune bucketization to achieve a desired level of privacy. Moreover, they propose a new method for answering range queries on multidimensional data [38]. Range queries over multiple attributes, e.g., $age < 20$ and $salary > 25$k, are allowed, while minimizing the cost of multidimensional bucketization. Yet, a threshold is defined to help the user control the trade-off between risk of data disclosure and cost.

4.2 Order-Preserving Indexing

Order Preserving Encryption Scheme (OPES) Agrawal et al.'s OPES is an OPE indexing scheme that supports range and equality queries over integers [4]. OPES transforms plaintexts with an order preserving function so that transformed values (e.g., index values) follow a target distribution. Comparison operations can be directly applied at the CSP's without inducing spurious tuples nor false positives. However, this scheme has been demonstrated to be vulnerable to statistical attacks [45].

OPE with Splitting and Scaling (OPESS) The OPESS scheme encrypts XML databases [72]. Wang et al. adopt splitting and scaling techniques to create index values following a uniform distribution. Plaintext order is preserved over indexes. Moreover, identical clear values are transformed into different indexes so that this scheme is robust against statistical attacks. However, this scheme flattens the frequency distribution of index values.

B+tree indexing Shmueli et al. [61] and Damiani et al. [21] use B+-trees built on database plaintext attribute values to preserve order in secure environments. B+-trees must either be stored in a trusted machine [36] or be encrypted at the CSP's, where each B+-tree is stored in a table with two attributes: node identifier and node content. In addition to ordering, B+-tree indexes support exact match, range and grouping queries, as well as predicates such as `LIKE`. For example, to execute a range query, the user sends a sequence of queries until reaching the leaf corresponding to the range's lower bound. Then, the node identifier helps retrieve all the tuples belonging to the range. The advantage of such indexing is that index content is not visible to the CSP and reveals no information about underlying plaintexts [21].

4.3 Searchable Encryption

SE allows the CSP to run keyword-based searches on encrypted data [63] that are particularly suitable to data outsourcing [16]. Considering a set of documents $\{D_i\}_{i=1,n}$ and an index of keywords $\{w_j\}_{j=1,m}$ describing the documents, users encrypt both documents D_i with any secure encryption scheme using a key

$\mathcal{K}_{Enc}$ and keywords w_j with a searchable scheme using a key $\mathcal{K}_{Index}$. The encrypted documents and index are then outsourced. When searching for documents containing some keywords, the user sends a so-called trapdoor encapsulating the keywords to the CSP [65]. Then, the CSP can search the encrypted index and the trapdoor to find the corresponding documents and send them back to the user. Both symmetric (private) and asymmetric (public) key encryption can be used to build symmetric SE (SSE) and asymetric SE (ASE) schemes, respectively [65]. ASE schemes support various query types such as range and subset queries, but are computationally intensive. SSE is more efficient than ASE, but supports fewer query types. SE induces a trade-off between security, efficiency and query expressiveness. SE schemes with higher levels of security induce higher complexity, while SE schemes supporting more query types are either less secure and/or less efficient [16]. Moreover, most SE schemes reveal access patterns, i.e., which documents contain a keyword. Only techniques based on PIR or ORAM do not.

PIR [17, 18, 47] enables a user to retrieve data from an outsourced database while preventing the CSP from learning any information about retrieved data [19], i.e., PIR enforces query privacy. Unfortunately, in a single-server setting, the only thing a user can do is retrieving the whole database, which induces communication overhead and annihilates the benefits of outsourcing. However, in a multiple-server setting where copies of the database are stored at k non-communicating/colluding CSPs, a user can hide queries by querying each server for a part of data, so that no server knows the whole query.

ORAM allows reading and writing to memory without revealing access patterns to the CSP [32]. In ORAM schemes, a user stores encrypted data at the CSP's and continuously shuffles and re-encrypts data as they are accessed [64]. Let $P = (q_1, \ldots, q_n)$ be an access pattern. The shuffling process induces the transformation of each query q_i into multiple queries, producing a new access pattern P'. An ORAM protocol is secure if two access patterns ORAM(P) and ORAM(P') are computationally indistinguishable.ORAM can be implemented using symmetric or fully homomorphic encryption (Section 5.1). An alternative solution for hiding access patterns is to frequently send fake queries to CSPsto prevent any adversary from inferring correlations between frequently queried data [48]. Yet, generating fake but realistic-looking queries is a challenge.

Unfortunately, a common limitation of PIR and ORAM schemes is a prohibitive query execution time [73].

4.4 Discussion

Table 2 provides a comparison of index-based methods with respect to the query types they allow and whether they require a post-processing step to eliminate false positives.

Table 2. Comparison of index-based methods

	Allowed queries			
	Exact match	Range	Aggregation	Post-processing
Hacigümüs et al. [33]	Yes	Yes	Yes	Yes
Mykletun & Tsudik [49]	No	No	Yes	No
Hore et al. [40]	Yes	Yes	No	Yes
Agrawal et al. [4]	Yes	Yes	No	No
Wang et al. [72]	Yes	Yes	No	No
Shmueli et al. [61], Damiani et al. [21]	Yes	Yes	No	No
Searchable Encryption [17, 18, 32, 47, 63, 65]	Yes	No	No	No

When defining an indexing method, it is important to consider two conflicting requirements. On one hand, the index should be related to the data well enough to allow efficient query execution. On the other hand, this relationship between plaintexts and the index should minimize the risk of any disclosure or loss of privacy [28, 59]. For example, in bucketization-based indexing, decreasing the number of buckets impairs performance, while a larger number of buckets increases the risk of data disclosure. In our database outsourcing scenario, a critical drawback of bucketization-based indexing is the loss of data granularity, which prevents grouping operations. The CSP can indeed not distinguish between tuples in buckets and the

user has to filter intermediate results sent by the CSP to reconstruct the global result. Hence, bucketization-based methods induce computational overhead at the user's, too. Such an overhead can be high, especially for queries that return a large number of encrypted tuples [70], e.g., grouping queries running on fine-grained data.

Finally, although index-based approaches are quite popular in cloud data outsourcing due to their efficiency [35, 59], their main limitation lies in data update. Typically, such methods but SE exploit the distribution of plaintexts, while update operations may change it, making index regeneration unavoidable [35]. As a result, index-based solutions but SE are suitable for read-only data. Yet, SE schemes are either too costly or too limited in query expressiveness to be used in practice.

5 Secure Databases

5.1 CryptDB

CryptDB is a pioneer system that allows efficient SQL query processing over ciphertexts into a DBMS [54]. The properties of a cryptographic scheme determine the kinds of queries that can be directly executed over ciphertexts. Thus, CryptDB implements several schemes with respect to different user-determined security requirements and query needs. Thus, we first describe these cryptographic schemes, and then we detail CryptDB's architecture.

Query-Aware Encryption Schemes

Random Encryption (RND) RND schemes are the strongest security schemes. They indeed guarantee semantic security, i.e., it is computationally impossible to distinguish two ciphertexts. For instance, let x be a plaintext value and E an RND encrypting function. If, using the same encryption key, $e_1 = E(x)$ and $e_2 = E(x)$, then with high probability, $e_1 \neq e_2$. However, RND schemes do not allow any computations nor queries over ciphertexts. They are only designed for safe storage.

Homomorphic Encryption (HE) HE allows performing arbitrary arithmetic operations over ciphertexts without decryption [31] while still providing semantic security. For instance, with an additive HE scheme, for any two encryptions $E(x)$ and $E(y)$, there exists a function f such that $f(E(x), E(y)) = E(x + y)$. Fully homomorphic encryption (FHE) is prohibitively slow and requires so much computing power that it cannot be used in practice as of today. However, partially homomorphic encryption (PHE) is efficient for specific operations and can be used in practice. PHE allows either addition or multiplication over ciphertexts and guarantees semantic security. Paillier's [53] and El Gamal's [27] are examples of PHE schemes. For instance, with Paillier's PHE, the product of two encryptions encrypts the sum of the encrypted values, i.e., $E(x) \times E(y) = E(x+y)$.

Deterministic Encryption (DET) DET encrypts identical data values into identical encryptions when using the same key, i.e., $\forall x, y$: $x = y \Leftrightarrow E(x) = E(y)$. Thus, DET allows queries with equality predicates, equijoins, as well as `GROUP BY`, `COUNT` and `DISTINCT` queries [55]. DET is secure only when there is no redundancy in data. It is not robust against statistical attacks. Although some public key encryption schemes allow exact match queries with stronger security guarantees [15], search takes linear time with the size of the database, while DET operates in logarithmic time [13], thus explaining its adoption in CryptDB.

Order Preserving Encryption (OPE) OPE is a deterministic encryption scheme that preserves plaintext order in ciphertexts. Let x and y be two plaintext values and E an OPE scheme. If $x \leq y$, then $E(x) \leq E(y)$. This feature allows range queries, `MIN` and `MAX` aggregations, and ordering over ciphertexts. In terms of security, OPE is weaker than DET because it reveals data order. Yet, it can provide sufficient security for some applications, e.g., when the adversary does not possess any prior knowledge, while increasing the efficiency of query processing [52].

Table 3 summarizes the features of the cryptographic schemes used in CryptDB.

Table 3. Features of CryptDB's encryption schemes

Allowed queries	RND	HE	DET	OPE
DISTINCT	No	No	Yes	Yes
WHERE (=, $\neq$)	No	No	Yes	Yes
Range queries	No	No	No	Yes
ORDER BY	No	No	No	Yes
JOIN	No	No	Yes	Yes
SUM, AVG	No	Yes	No	No
MIN, MAX	No	No	No	Yes
GROUP BY	No	No	Yes	Yes
Information leakage	None	None	Duplicates	Data order

CryptDB's Architecture CryptDB follows three principles to solve the problem of querying encrypted databases: 1) *SQL-aware encryption* that uses cryptographic schemes within SQL queries; 2) *adjustable query-based encryption* to minimize data leakage; and 3) *chain cryptographic keys in user passwords* to enable data decryption only for authorized users with access privileges.

In CryptDB's core, encryption is structured in multiple embedded levels akin to onion layers. Each onion layer helps process given classes of queries. The outermost layers are RND and HE, HE actually being Paillier's PHE scheme. RND and HE provide the highest level of security, whereas inner layers, OPE and DET, provide more functionality. The OPE layer is an enhancement of [14]. Eventually, two new cryptographic schemes enable join operations.

Ciphertext access is achieved through a trusted proxy server that encrypts data, rewrites queries (by anonymizing table and attribute names and encrypting constants) and decrypts query results. The proxy server stores encryption keys, the database schema and the onion layers of all attributes in the database. When a query is issued, the proxy dynamically peels off onion layers downs to a layer corresponding to the given computation. For instance, consider the query `SELECT * FROM employee WHERE name = 'Alice'`. First, the proxy issues a query to peel off the RND layer for attribute `name` down to the DET layer. Then, the proxy rewrites the query as `SELECT * FROM T1 WHERE A2 = '0xac18f'`, where `T1` and `A2` denote the anonymization of table `employee` and attribute `name`, respectively, and `0xac18f` $= E_{DET}$('Alice'). Similary, aggregation query `SELECT SUM(salary) FROM employee` would translate as `SELECT` $\texttt{SUM}_{HE}$`(A3) FROM T1`, where $\texttt{SUM}_{HE}$ is a user-defined function implementing Paillier's PHE and `A3` is the anonymization of attribute `salary`.

5.2 MONOMI

While CryptDB offers one of the first practical solutions for secure DBMSs, there are still a lot of queries that are not supported, especially OLAP-like queries. As an illustration, CryptDB supports only 2 queries out of 22 from the TPC-H decision support benchmark [69]. Thence, MONOMI builds upon CryptDB to allow the execution of analytical workloads [70].

To this aim, MONOMI adds in a designer that optimizes the physical database layout at the CSP's and a query planner that splits query execution between the CSP and the user. The optimal plan for executing some queries may indeed involve sending intermediate results between the user and the CSP several times to execute different parts of a query [70]. For instance, to run a `SUM / GROUP BY / HAVING` query, MONOMI computes the `SUM` and `GROUP BY` at the CSP's through the HE and DET encryption schemes, respectively. Then, since HE does not preserve data order, the `HAVING` statement is executed at the user's after decryption. This strategy helps MONOMI allow 19 out of the 22 queries of TPC-H.

5.3 Multi-Valued Order Preserving Encryption (MV-OPE)

Lopes et al. rightly claim that "little attention has been devoted to determine how a data warehouse hosted in a cloud should be encrypted to enable analytical queries processing" [46]. Thence, they propose the MV-OPE scheme that allows `GROUP BY` queries over ciphertexts. Such a scheme could replace CryptDB's and MONOMI's OPE without having to compute anything at the user's.

Generally speaking, MV-OPE extends OPE by encrypting the same plaintext into different ciphertexts while preserving the order of the plaintexts [41]. Thus, given two clear values x and y and an MV-OPE function E, if $x < y$ then $E(x) < E(y)$. MV-OPE can be used to compute operations such as equality, difference, inequalities, minimum, maximum and count [46]. MV-OPE improves robustness against statistical attacks and only leaks the order of data. Lopes et al.'s scheme combines MV-OPE with FHE (Section 5.1). Moreover, as CryptDB and MONOMI, it involves a secure host, e.g., a trusted proxy server. Despite using FHE, Lopes et al. experimentally show that computing queries over ciphertexts at the CSP's is significantly faster than computing them at the user's after decryption.

5.4 Secure Trusted Hardware

Trusted hardware devices are widely used for security, e.g., smart cards for secure authentication and secure coprocessors in automated teller machines (ATMs). Quite naturally, the idea of processing queries inside tamper-proof enclosures of trusted hardware, such as a secure coprocessor or Field Programmable Gate Array (FPGA)-based secure programmable hardware [26], came up. Such components are physically hosted at the CSP's. They have access to encryption keys and allow performing a limited set of queries over ciphertexts.

TrustedDB TrustedDB is an SQL database processing engine that makes use of IBM 4764/5 cryptographic coprocessors [10] to run custom queries securely [12]. Coprocessors offer several cryptographic schemes such as the Advanced Encryption Standard (AES), the Triple Data Encryption Standard (3DES), RSA, pseu-do-random number generation and cryptographic hash functions. Yet, cryptographic coprocessors are significantly constrained in both computation ability and memory capacity. Thus, a trade-off must be considered between cheap query processing on untrusted main processors (at the CSP's) and expensive computation inside secure coprocessors.

Sensitive data can only be decrypted and processed by the user or a secure coprocessor. Only non-sensitive data are stored unencrypted at the CSP's. When a query is issued, it is encrypted at the user's, rewritten as a set of subqueries and executed at the CSP's or in the secure coprocessor database engine, with respect to data sensitivity. The final result is assembled, encrypted by the secure coprocessor and sent back to the user.

Cipherbase Cipherbase aims at deploying trusted hardware for secure data processing in the cloud [9]. Cipherbase actually extends Microsoft SQL Server with in-server, customized FPGA-based trusted hardware. The FPGA is a trusted black box for computing operations over ciphertexts, which are encrypted with a non-homomorphic encryption scheme such as AES. The FPGA decrypts data internally, processes the operations and encrypts the result back. As in TrustedDB, query processing on non-sensitive data is handled by the CSP.

5.5 Discussion

CryptDB is much cited, but is quite insecure and introduces some loopholes. Its onion adjustable encryption architecture is indeed unidirectional, i.e., once an attribute is set down to a weak scheme such as DET, it never returns to a higher encryption level [43]. Moreover, attributes targeted by exact match and range queries are encrypted with DET and OPE, respectively, and are vulnerable to statistical attacks. As a result, once an exact match or range query is issued, the system becomes vulnerable ever after. DET and OPE have even been shown to be much more insecure than previously expected [50]. Additionally, peeling down onion layers induces an overhead, especially in the case of big tables.

Moreover, although CryptDB does support many types of queries, there are still many unsupported types of queries, e.g., predicate evaluation on more than one attribute. MONOMI addresses this shortcoming, but retains the same security mechanisms as CryptDB. MONOMI also induces a heavy communication overhead between the user and the CSP, since intermediate results may be exchanged several times to execute different parts of a query [70].

Despite a distributed architecture, Lopes et al's solution requires a trusted server to securely execute `GROUP BY` queries. In our database outsourcing scenario, all service providers that are external to the user's

are considered untrusted. Thus, Lopes et al's trusted server would be located at the user's, inducing costs that do not fit our scenario. Additionally, this solution does not support `MIN` and `MAX` aggregation operators directly over ciphertexts.

Finally, beside computation ability and memory capacity limitations, trusted hardware is still very expensive, which is again contrary to our scenario that aims at using cheap commodity machines in the cloud. Moreover, leaving unencrypted attributes jeopardizes ciphertext, because relationships between ciphertexts and plaintexts may reveal information about ciphertexts [8].

6 Conclusion

Although encryption methods enforce privacy, in some cases, the impact on performance makes them inapplicable to cloud databases. It is indeed currently impossible to develop a system that meets both state-of-the-art cryptographic security standards and query performance requirements. In this final section, we provide a global discussion on security, performance and storage requirements for secure databases, before concluding the paper.

6.1 Security

The DET and OPE schemes, which are notably used in CryptDB, allow efficiently performing queries over ciphertexts. Database optimization techniques, e.g., usual indexing methods, can also be used to enhance query performance. However, DET and OPE leak a non-negligible amount of information and are vulnerable to statistical attacks [42]. For example, a large fraction of tuples from DET encrypted attributes can be decrypted by statistical attacks [50]. The vulnerability of DET is extremely detrimental to DBs with high redundancy, e.g., data warehouses. The weak security of OPE makes it inappropriate, too. It is indeed even worse than DET in terms of security [29, 42]. Eventually, a recent class of generic attacks against private range query schemes invalidates much of the existing literature [42].

Thus, FHE looks like a more appropriate choice for encryption. In particular, PHE encryption can be used to sum ciphertexts, but the cost of decryption at the client's can remain high. As of today, it is indeed usually more efficient to decrypt data at the client's and then perform the aggregation, rather than processing aggregation queries over ciphertexts at the CSP's [70]. Yet, FHE is likely to become a viable alternative in the upcoming decade, with both new FHE schemes and improvements in hardware performance. However, since preserving the order of data is necessary when running queries such as sorting, grouping and range operations, the issue of designing order preserving FHE schemes will have to be addressed.

6.2 Query Post-Processing

Tuple and table-level encryption are casually considered preferable to attribute-level encryption, because of lower startup costs at the user's and minimal storage costs at the CSP's [39]. However, the loss of data granularity is an important deficiency in scenarios such as OLAP. Thus, some solutions that use tuple-level encryption (Section 4) handle query processing by means of auxiliary indexes at the CSP's (e.g., bucketization-based indexing) and perform final query processing at the user's. Similarly, MONOMI splits the execution of queries between the user and CSP. In such solutions, it is essential to cut down the bandwidth required to transfer intermediate results and user computational resources for user side query processing [70], which is quite an open issue. CPU and storage usage at the user's must indeed be minimum for maintaining the benefits of outsourcing.

6.3 Storage Overhead

CryptDB, MONOMI and Cipherbase use attribute-level encryption, i.e., each attribute value is encrypted independently [9], at the cost of storage overhead. For instance, using classical AES in Cipher-Block Chaining (CBC) mode, a 32-bit integer is encrypted on 256 bits [9]. Worse, Paillier's PHE scheme, which is used in CryptDB, operates over 2048-bit ciphertext [70]. MONOMI addresses this issue by packing multiple values from a single tuple into one PHE encryption, using Ge and Zdonik's scheme [30]. This optimization

works properly for a table with many PHE-encrypted attributes, but would complicate partial updates that reset some but not all attribute values packed into a PHE tuple encryption [55]. Thus, although security vs. performance is necessarily a tradeoff, there is still some room for improving the storage overhead of cryptographic schemes, especially for secret sharing schemes.

6.4 Computational Overhead

Operations at the CSP's should not involve any expensive arithmetic operations such as modular multiplication or exponentiation [62]. However, for instance in Paillier's scheme, encrypting the sum of two clear values x and y requires multiplying ciphertexts $E(x)$ and $E(y)$ modulo a 2048-bit public key, i.e., $E(x + y) = E(x) \times E(y)$. Such modular multiplications are computationally expensive, especially on big tables.

MONOMI implements a grouped homomorphic addition optimization. All to-be-aggregated attributes are packed in such a way that aggregation queries can be computed with a single modular multiplication. This implies that all queries must be declared ahead of time, which it is not possible for all applications, e.g., OLAP ad-hoc navigation. Yet, performance optimization techniques, such as indexing, partitioning or view materialization, can apply onto ciphertexts. However, although they speed up some queries, they also slow down others [70]. As a result, it is crucial to select a cryptographic method that meets all usage constrains. Again, a tradeoff must be defined to meet the intended level of privacy while minimizing the impact on performance.

6.5 Wrap-up

In this paper, we review the security mechanisms that can nowadays be used in the deployment of cloud databases. We particularly focus on the cryptographic schemes and the (would-be) secure systems that enable executing queries over ciphertexts without decryption. This survey highlights the potential benefits of existing solutions in a cloud computing context, but also that one must take great care about security guarantees before selecting one such solution.

Moreover, cryptography cannot prevent all attacks by malicious adversaries, e.g., Distributed Denial of Service (DDoS) attacks. It is thus essential to clearly specify the objectives of cloud database deployment, to adopt security mechanisms that are adapted to these objectives. Such preliminary work shall determine the initialization of secure protocols, the choice of cryptographic schemes, the need for a trusted third party, etc.

Finally, since computational performance is currently still a bottleneck, resorting to data distribution and query parallelization must be a priority. Thus, cloud frameworks such as Hadoop [7] and Spark [6] should be exploited in future secure cloud DBMSs.

References

1. Charu C. Aggarwal and Philip S. Yu. A General Survey of Privacy-Preserving Data Mining Models and Algorithms. In *Privacy-Preserving Data Mining: Models and Algorithms*, pages 11–52. Springer, 2008.
2. Gagan Aggarwal, Mayank Bawa, Prasanna Ganesan, Hector Garcia-Molina, Krishnaram Kenthapadi, Rajeev Motwani, Utkarsh Srivastava, Dilys Thomas, and Ying Xu. Two Can Keep A Secret: A Distributed Architecture for Secure Database Services. In *2^{nd} Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA*, pages 186–199, 2005.
3. Divyakant Agrawal, Amr El Abbadi, Fatih Emekçi, and Ahmed Metwally. Database Management as a Service: Challenges and Opportunities. In *25^{th} International Conference on Data Engineering (ICDE), Shanghai, China*, pages 1709–1716, 2009.
4. Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order-Preserving Encryption for Numeric Data. In *ACM SIGMOD International Conference on Management of Data (SIGMOD), Paris, France*, pages 563–574, 2004.
5. Rakesh Agrawal, Ramakrishnan Srikant, and Dilys Thomas. Privacy Preserving OLAP. In *ACM SIGMOD International Conference on Management of Data (SIGMOD), Baltimore, MD, USA*, pages 251–262, 2005.
6. Apache Software Foundation. Apache Spark – Lightning-fast cluster computing. `https://spark.apache.org`, 2016.

7. Apache Software Foundation. Hadoop. `http://hadoop.apache.org`, 2016.
8. Arvind Arasu, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravishankar Ramamurthy, and Ramarathnam Venkatesan. Orthogonal Security with Cipherbase. In *6th Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA*, 2013.
9. Arvind Arasu, Ken Eguro, Manas Joglekar, Raghav Kaushik, Donald Kossmann, and Ravi Ramamurthy. Transaction processing on confidential data using Cipherbase. In *31st IEEE International Conference on Data Engineering (ICDE), Seoul, Korea*, pages 435–446, 2015.
10. Todd W. Arnold, Carl U. Buscaglia, F. Chan, Vincenzo Condorelli, John C. Dayka, W. Santiago-Fernandez, Nihad Hadzic, Michael D. Hocker, M. Jordan, T. E. Morris, and Klaus Werner. IBM 4765 cryptographic coprocessor. *IBM Journal of Research and Development*, 56(1):10, 2012.
11. Varunya Attasena, Nouria Harbi, and Jérôme Darmont. fVSS: A New Secure and Cost-Efficient Scheme for Cloud Data Warehouses. In *17th International Workshop on Data Warehousing and OLAP (DOLAP), Shanghai, China*, pages 81–90, 2014.
12. Sumeet Bajaj and Radu Sion. TrustedDB: a trusted hardware based database with privacy and data confidentiality. In *ACM SIGMOD International Conference on Management of Data (SIGMOD), Athens, Greece*, pages 205–216, 2011.
13. Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and Efficiently Searchable Encryption. In *27th Annual International Cryptology Conference (CRYPTO), Santa Barbara, CA, USA*, pages 535–552, 2007.
14. Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-Preserving Symmetric Encryption. In *28th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Cologne, Germany*, pages 224–241, 2009.
15. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public Key Encryption with Keyword Search. In *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Interlaken, Switzerland*, pages 506–522, 2004.
16. Christoph Bösch, Pieter H. Hartel, Willem Jonker, and Andreas Peter. A Survey of Provably Secure Searchable Encryption. *ACM Computing Surveys*, 47(2):18:1–18:51, 2014.
17. Christian Cachin, Silvio Micali, and Markus Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. In *International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYT), Prague, Czech Republic*, pages 402–414, 1999.
18. Yan-Cheng Chang. Single Database Private Information Retrieval with Logarithmic Communication. In *9th Australasian Conference on Information Security and Privacy (ACISP), Sydney, Australia*, pages 50–61, 2004.
19. Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private Information Retrieval. *Journal of the ACM*, 45(6):965–981, 1998.
20. Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Selective data outsourcing for enforcing privacy. *Journal of Computer Security*, 19(3):531–566, 2011.
21. Ernesto Damiani, Sabrina De Capitani di Vimercati, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In *10th ACM Conference on Computer and Communications Security (CCS), Washington, DC, USA*, pages 93–102, 2003.
22. George I. Davida, David L. Wells, and John B. Kam. A Database Encryption System with Subkeys. *ACM Transactions on Database Systems*, 6(2):312–328, 1981.
23. Yves-Alexandre de Montjoye, Csar A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. Unique in the Crowd: The privacy bounds of human mobility. Nature Scientific Reports 3, Article number: 1376, `http://www.nature.com/articles/srep01376`, 2013.
24. Cynthia Dwork. Differential Privacy. In *33rd International Colloquium on Automata, Languages and Programming (ICALP), Venice, Italy*, volume 4052 of *LNCS*, pages 1–12, 2006.
25. Cynthia Dwork. Differential Privacy. In *Encyclopedia of Cryptography and Security*, pages 338–340. Springer, 2011.
26. Ken Eguro and Ramarathnam Venkatesan. FPGAs for trusted cloud computing. In *22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway*, pages 63–70, 2012.
27. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
28. Yuval Elovici, Ronen Waisenberg, Erez Shmueli, and Ehud Gudes. A Structure Preserving Database Encryption Scheme. In *Secure Data Management Workshop (SDM), Toronto, Canada*, pages 28–40, 2004.
29. Jun Furukawa. Short Comparable Encryption. In *13th International Conference in Cryptology and Network Security (CANS), Heraklion, Crete, Greece*, pages 337–352, 2014.
30. Tingjian Ge and Stanley B. Zdonik. Answering Aggregation Queries in a Secure System Model. In *33rd International Conference on Very Large Data Bases (VLDB), Vienna, Austria*, pages 519–530, 2007.
31. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

32. Oded Goldreich and Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.
33. Hakan Hacigümüs, Balakrishna R. Iyer, Chen Li, and Sharad Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *ACM SIGMOD International Conference on Management of Data (SIGMOD), Madison, WI, USA*, pages 216–227, 2002.
34. Hakan Hacigümüs, Balakrishna R. Iyer, and Sharad Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In *9^{th} International Conference ob Database Systems for Advances Applications (DASFAA), Jeju Island, Korea*, pages 125–136, 2004.
35. Mohammad Ali Hadavi, Ernesto Damiani, Rasool Jalili, Stelvio Cimato, and Zeinab Ganjei. AS5: A Secure Searchable Secret Sharing Scheme for Privacy Preserving Database Outsourcing. In *7^{th} International Workshop on Data Privacy Management (DPM), Pisa, Italy*, pages 201–216, 2012.
36. Mohammad Ali Hadavi and Rasool Jalili. Secure data outsourcing based on threshold secret sharing; towards a more practical solution. In *36^{th} International Conference on Very Large Data Bases (VLDB) PhD Workshop, Singapore*, pages 54–59, 2010.
37. Mohammad Ali Hadavi, Morteza Noferesti, Rasool Jalili, and Ernesto Damiani. Database as a Service: Towards a Unified Solution for Security Requirements. In *36^{th} Annual IEEE Computer Software and Applications Conference (COMPSAC) Workshops, Izmir, Turkey*, pages 415–420, 2012.
38. Bijit Hore, Sharad Mehrotra, Mustafa Canim, and Murat Kantarcioglu. Secure multidimensional range queries over outsourced data. *VLDB Journal*, 21(3):333–358, 2012.
39. Bijit Hore, Sharad Mehrotra, and Hakan Hacigümüç. Managing and querying encrypted data. In *Handbook of Database Security*, pages 163–190. Springer, 2008.
40. Bijit Hore, Sharad Mehrotra, and Gene Tsudik. A Privacy-Preserving Index for Range Queries. In *30^{th} International Conference on Very Large Data Bases (VLDB), Toronto, Canada*, pages 720–731, 2004.
41. Hasan Kadhem, Toshiyuki Amagasa, and Hiroyuki Kitagawa. MV-OPES: Multivalued-Order Preserving Encryption Scheme: A Novel Scheme for Encrypting Integer Value to Many Different Values. *IEICE Transactions on Information and Systems*, 93-D(9):2520–2533, 2010.
42. Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic Attacks on Secure Outsourced Databases. In *23^{rd} ACM Conference on Computer and Communications Security (CCS), Vienna, Austria*, pages 1329–1340, 2016.
43. Florian Kerschbaum, Patrick Grofig, Isabelle Hang, Martin Härterich, Mathias Kohler, Andreas Schaad, Axel Schröpfer, and Walter Tighzert. Adjustably encrypted in-memory column-store. In *ACM SIGSAC Conference on Computer and Communications Security (CCS), Berlin, Germany*, pages 1325–1328, 2013.
44. Hugo Krawczyk. Secret Sharing Made Short. In *13^{th} Annual International Cryptology Conference (CRYPTO), Santa Barbara, CA, USA*, pages 136–146, 1993.
45. Zheli Liu, Xiaofeng Chen, Jun Yang, Chunfu Jia, and Ilsun You. New order preserving encryption model for outsourced databases in cloud environments. *Journal of Network and Computer Applications*, 59:198–207, 2016.
46. Claudivan Cruz Lopes, Valéria Cesário Times, Stan Matwin, Ricardo Rodrigues Ciferri, and Cristina Dutra de Aguiar Ciferri. Processing OLAP Queries over an Encrypted Data Warehouse Stored in the Cloud. In *16^{th} International Conference on Data Warehousing and Knowledge Discovery (DaWaK), Munich, Germany*, pages 195–207, 2014.
47. Wouter Lueks and Ian Goldberg. Sublinear Scaling for Multi-Client Private Information Retrieval. In *19^{th} International Conference on Financial Cryptography and Data Security (FC), San Juan, Puerto Rico*, pages 168–186, 2015.
48. Charalampos Mavroforakis, Nathan Chenette, Adam O'Neill, George Kollios, and Ran Canetti. Modular Order-Preserving Encryption, Revisited. In *ACM SIGMOD International Conference on Management of Data, Melbourne, Australia*, pages 763–777, 2015.
49. Einar Mykletun and Gene Tsudik. Aggregation Queries in the Database-As-a-Service Model. In *20^{th} Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec), Sophia Antipolis, France*, pages 89–103, 2006.
50. Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference Attacks on Property-Preserving Encrypted Databases. In *22^{nd} ACM SIGSAC Conference on Computer and Communications Security (CCS), Denver, CO, USA*, pages 644–655, 2015.
51. Oracle Corporation. Data Masking Best Practices. White paper, 2013.
52. Gultekin Özsoyoglu, David A Singer, and Sun S Chung. Anti-Tamper Databases: Querying Encrypted Databases. In *17^{th} Annual IFIP WG 11.3 Working Conference on Data and Application Security (DBSec), Estes Park, CO, USA*, pages 133–146, 2003.
53. Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT), Prague, Czech Republic*, pages 223–238, 1999.

54. Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: protecting confidentiality with encrypted query processing. In *23rd ACM Symposium on Operating Systems Principles (SOSP), Cascais, Portugal*, pages 85–100, 2011.
55. Raluca Ada Popa. *Building practical systems that compute on encrypted data*. PhD thesis, Massachusetts Institute of Technology, 2014.
56. Sara Ricci, Josep Domingo-Ferrer, and David Sánchez. Privacy-Preserving Cloud-Based Statistical Analyses on Sensitive Categorical Data. In *13th International Conference on Modeling Decisions for Artificial Intelligence (MDAI), Sant Julià de Lòria, Andorra*, volume 9880 of *LNAI*, pages 227–238, 2016.
57. Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation*, 4(11):169–180, 1978.
58. Eyad Saleh, Ahmad Alsa'deh, Ahmad Kayed, and Christoph Meinel. Processing Over Encrypted Data: Between Theory and Practice. *SIGMOD Record*, 45(3):5–16, 2016.
59. Pierangela Samarati and Sabrina De Capitani di Vimercati. Data Protection in Outsourcing Scenarios: Issues and Directions. In *5th ACM Symposium on Information, Computer and Communications Security (ASIACCS), Beijing, China*, pages 1–14, 2010.
60. Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.
61. Erez Shmueli, Ronen Waisenberg, Yuval Elovici, and Ehud Gudes. Designing secure indexes for encrypted databases. In *19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec), Storrs, CT, USA*, pages 54–68, 2005.
62. Radu Sion. Towards Secure Data Outsourcing. In *Handbook of Database Security – Applications and Trends*, pages 137–161. Springer, 2008.
63. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical Techniques for Searches on Encrypted Data. In *IEEE Symposium on Security and Privacy (SP), Berkeley, CA, USA*, pages 44–55, 2000.
64. Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. In *ACM SIGSAC Conference on Computer and Communications Security (CCS), Berlin, Germany*, pages 299–310, 2013.
65. Wenhai Sun, Wenjing Lou, Y. Thomas Hou, and Hui Li. Privacy-Preserving Keyword Search Over Encrypted Data in Cloud Computing. In *Secure Cloud Computing*, pages 189–212. Springer, 2014.
66. Sam Yuan Sung, Yao Liu, Hui Xiong, and Peter A. Ng. Privacy preservation for data cubes. *Knowledge and Information Systems*, 9(1):38–61, 2006.
67. Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.
68. Brian Thompson, Stuart Haber, William G. Horne, Tomas Sander, and Danfeng Yao. Privacy-Preserving Computation and Verification of Aggregate Queries on Outsourced Databases. In *9th International Symposium on Privacy Enhancing Technologies (PETS), Seattle, WA, USA*, pages 185–201, 2009.
69. Transaction Performance Processing Council. TPC Benchmark H (Decision Support) Standard Specification Revision 2.1. `http://www.tpc.org`, 2014.
70. Stephen Tu, M. Frans Kaashoek, Samuel Madden, and Nickolai Zeldovich. Processing Analytical Queries over Encrypted Data. *Proceedings of the VLDB Endowment*, 6(5):289–300, 2013.
71. Shiyuan Wang, Divyakant Agrawal, and Amr El Abbadi. A Comprehensive Framework for Secure Query Processing on Relational Data in the Cloud. In *Secure Data Management Workshop (SDM), Seattle, WA, USA*, pages 52–69, 2011.
72. Wendy Hui Wang and Laks V. S. Lakshmanan. Efficient Secure Query Evaluation over Encrypted XML Databases. In *32nd International Conference on Very Large Data Bases, Seoul, Korea*, pages 127–138, 2006.
73. Peter Williams and Radu Sion. Access privacy and correctness on untrusted storage. *ACM Transactions on Information and System Security*, 16(3):12, 2013.
74. Xiaokui Xiao, Gabriel Bender, Michael Hay, and Johannes Gehrke. iReduct: differential privacy with reduced relative errors. In *ACM SIGMOD International Conference on Management of Data (SIGMOD), Athens, Greece*, pages 229–240, 2011.
75. Li Xiong, Subramanyam Chitti, and Ling Liu. Preserving data privacy in outsourcing data aggregation services. *ACM Transactions on Internet Technology*, 7(3):17, 2007.
76. Yin Yang, Zhenjie Zhang, Gerome Miklau, Marianne Winslett, and Xiaokui Xiao. Differential privacy in data publication and analysis. In *ACM SIGMOD International Conference on Management of Data, Scottsdale, AZ, USA*, pages 601–606, 2012.
77. Noel Yuhanna, Mike Gilpin, and Adam Knoll. Your Enterprise Database Security Strategy 2010. Forrester – `http://www.oracle.com/us/ciocentral/forrester-database-security-396253.pdf`, 2009.