IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 9, 2019, accepted March 19, 2019, date of publication April 1, 2019, date of current version April 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2908761

Toward Securing Cloud-Based Data Analytics: A Discussion on Current Solutions and Open Issues

SOMAYEH SOBATI MOGHADAM[1] AND AMJAD FAYOUMI[2], (Member, IEEE)
[1]Department of Computer Engineering, Faculty of Electrical and Computer Engineering, University of Hakim Sabzevari, Sabzevar, Iran
[2]Management Science, Lancaster University Management School, Lancaster LA1 4YX, U.K.

Corresponding author: Somayeh Sobati Moghadam (s.sobati@ hsu.ac.ir)

ABSTRACT In the last few years, organizations and business professionals have realized the value of collaborative data analytics in supporting decision-making. Where several activities are performed on online data by different stakeholders, such as cleansing, aggregation, analysis, and visualization, cloud-based data analytics has become a favored choice for business professionals due to the elasticity, availability, scalability, and pay-as-you-go features offered by cloud computing. However, large amounts of data stored on the cloud are very sensitive (e.g., innovation, financial, legal, and customers' data), and so data privacy remains one of the top concerns for many reasons; mainly those relating to legal or competition issues. In this paper, we review the security and cryptographic mechanisms which aim to make data analytics secure in a cloud environment and discuss current research challenges.

INDEX TERMS Cloud computing, data analytics, data privacy, query processing.

I. INTRODUCTION

With cloud-based technologies and services flourishing, many organizations have started adopting hybrid Information System (IS) solutions, particularly the multi-hybrid cloud deployment model, where IS services are outsourced and shared with several cloud service providers and integrated with the organization's on-premises systems. Cloud computing appeals to businesses and organizations because of the wide variety of benefits it offers. The pay-per-use model of cloud computing is very attractive, especially for small and medium enterprises, because it reduces start-up costs [95]. Some cloud outsourcing models go as far as outsourcing a complete business process to a third party Cloud Service Provider (CSP), and they share part of their IS assets through the cloud system [4].

The growth of cloud-based services has made data analytics a feasible reality for organizations [95]. One closely related outsourcing model that has recently emerged is data analytics outsourcing [5], where organizations offer a value-added third party agent access to their heterogeneous, large datasets stored on the cloud in order to perform some analytical tasks and deliver insight to the organization [6]. Data analytics helps businesses and organizations analyze data to gain a fuller understanding of the situation confronting them and make better decisions [7]. Data analytics outsourcing services, or Data Analytics-as-a-Service (DAaaS), entails storing sensitive data on a remote CSP's infrastructure, and querying and retrieving data on demand [8]. Data Analytics-as-a-Service is a customizable analytical platform which uses the software-as-a-service (SaaS) cloud-based service delivery model. Different customizable data analytics tools are typically used by companies which are collaborating in an integrated value chain. The companies use DAaaS with the intention of integrating cross-organizational business processes to maximize data-driven value and enable rapid innovation.

The security solutions and challenges we discuss in this paper relate to securing data which is stored on a CSP's infrastructure and is subject to limited access by multiple organizations. These organizations are collaborating with each other, and data analytics is crucial for them collectively.

Although cloud data outsourcing provides many benefits, it also raises security and privacy concerns. Ensuring data privacy is not a trivial matter, because data owners lose control of their data in cloud environments. Threats to privacy come from both outside and inside, so data privacy must be guaranteed not only against external adversaries breaking into

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Hui Yeh.

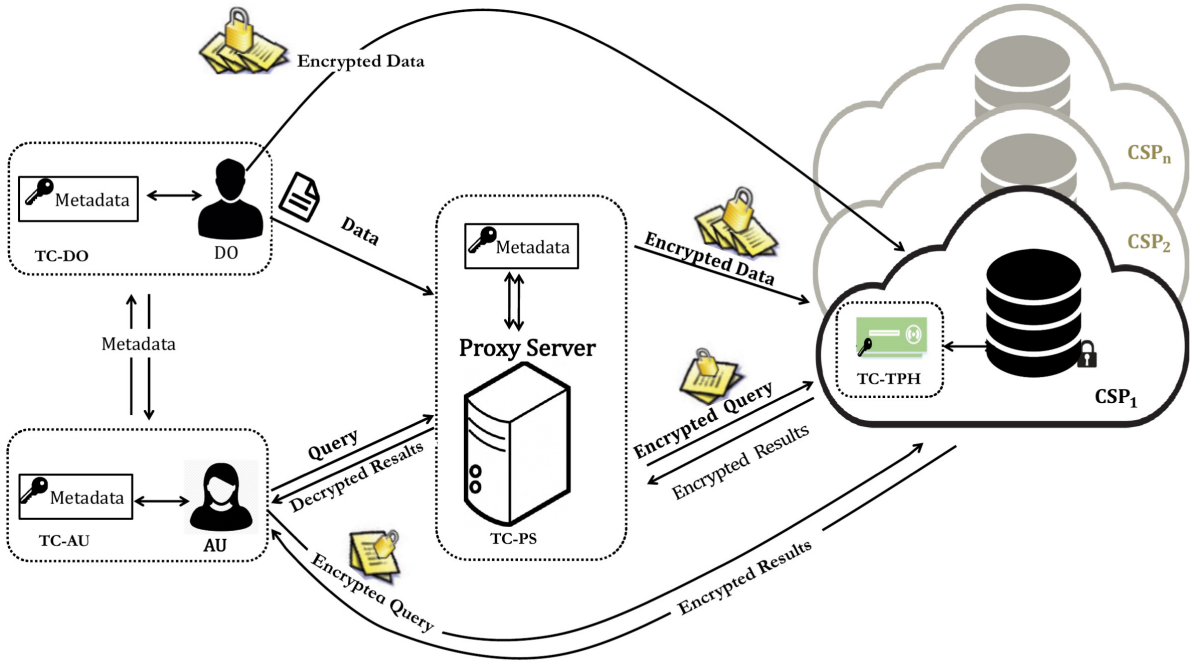


FIGURE 1. System architecture for securing cloud-based data analytics.

the system, but also malicious insiders and so-called honest but curious CSPs [9].

A straightforward solution to the problem of data privacy is to encrypt sensitive data locally in a trusted environment before sending it to a CSP. Encryption protects sensitive data even if the server is compromised. However, encrypting data incurs additional storage costs, and query processing over encrypted data is both non-trivial and costly in terms of computing power in the cloud, and thus money. Moreover, in addition to overheads, the levels of security and functionality are other issues which must be addressed properly. Typically, existing solutions from the literature [10]–[13] cannot provide a seamless and complete solution which guarantees secure cloud data analytics, because they fail to deal adequately with at least one of these issues [14]. Searchable Encryption (SE) allows keyword searches over encrypted documents [87], [88]. However, SE schemes involve a trade-off between security and efficiency.

A common limitation of SE schemes is their prohibitive computational cost. In the context of cloud outsourcing, access control can be achieved by using attribute-based encryption (ABE), which enables the decryption of data when a user has certain attributes which satisfy the access structure [89]. However, the main disadvantages of ABE schemes are deficiency and functionality issues. Consequently, choosing an efficient cryptographic scheme with adequate levels of security and functionality for computing analytical queries over encrypted data, which would thus be an effective solution for secure cloud-based data analytics, is still a challenge.

In this paper, we give a comprehensive overview of secure cloud-based data analytics which provides an easy entry point for researchers with no cryptographic background. We introduce a general architecture and system model which will accommodate a wide range of uses of cloud data analytics. This architecture incorporates a specific adversary and security model, and we provide an overview of cryptographic tools for ensuring data privacy with an emphasis on functionality and efficiency in terms of overheads. We then present and compare practical secure solutions from the literature and identify various security and efficiency issues. A comparative analysis of security, overhead and functionality issues is presented at the end of the article in order to provide a series of guidelines and insights to assist in designing and developing secured cloud-based data analytics with a level of functionality and efficiency which meets user requirements. The rest of the paper is organized as follows: we present our reference system architecture, adversary and security model for cloud data analytics in Section II. We classify the various cryptographic schemes which can be implemented in the reference system for preserving privacy in Section n III. We then review, discuss and compare practical solutions in Section IV, while limitations and unresolved issues are discussed in Section V. Finally, conclusions are drawn in Section VI.

II. ASSUMPTIONS AND REFERENCE MODEL

In this section, we introduce the main actors and their activities in relation to data analytics in the cloud. Then, we describe the ways in which the security of cloud-based data analytics can be compromised. Since we review cryptographic approaches for preserving privacy, we define a standard security model to specify the level of security guarantees desired in cloud-based data analytics. Taking an adversary

and security model as our basis, we elaborate the system goals which must be achieved in the cloud context

A. ACTORS AND ACTIVITIES IN CLOUD DATA ANALYTICS

A cloud-based data analytics system allows a server to store and query encrypted data on behalf of a user without gaining information about the underlying data.

Figure 1 depicts the basic system model and architecture of cloud-based data analytics, which consists of three main entities: the Data Owner, the Cloud Service Provider and Authorized Users.

The **Data Owner (DO)** is an entity which has a large amount of data to be outsourced in the cloud, and can be an individual user, or a large, small or medium business or enterprise.

The **Cloud Service Provider (CSP)** provides data storage services and computational resources.

Authorized Users (AU): the DO allows the authorized users, which could be other enterprises or individual users, to use the outsourced data. AU can query encrypted data and retrieve encrypted results and decrypt them to get corresponding plaintext.

In a secure cloud-based architecture, a trusted component (TC) must execute confidential tasks such as key management functions, query rewriting and post-processing, and decryption. The DO and the AU are assumed the TC in some settings (TC-DO and TC-AU in Figure 1). In this setting, the TC must be installed and maintained on each AU and DO separately. To avoid installing the TC in several machines, the TC is set between the client (i.e., the DO or the AU) and the CSP [12], which is called the proxy server (TC-PS in Figure 1). In this setting, the proxy server intercepts client/server communications and performs data/query encryption and decryption on behalf of the user. The proxy server maintains some metadata and the private keys for data encryption and query transformation. Finally, tamper-proof hardware can be embedded at the CSP (TC-TPH in Figure 1), which performs encryption/decryption and query rewriting [15]. The AU also needs to perform a few security tasks like query rewriting and decryption. Typically, the interactions in this architecture are as follows:

1) The DO outsources data to a CSP (or multiple CSPs [16], [17], which are non-colluding with each other or with other external adversaries) in encrypted form while still maintaining the capability to query the data efficiently.

2) An AU encrypts a query and sends the encrypted query to the CSP.

3) The CSP executes the query over encrypted data and returns encrypted results to the AU.

4) Finally, the AU decrypts the results.

All interactions between the DO/AU and the CSP can be executed through the proxy server, TC-PS, when the TC-PS is the TC.

B. ADVERSARY MODEL

An adversary model specifies an adversary's ability to threaten security. In cloud outsourcing scenarios, the adversary can be either honest but curious (passive), or malicious (active). Honest but curious is a widely-used adversary model for cloud outsourcing scenarios [10], [11], [18]–[21]. An honest but curious CSP or insider faithfully complies with any service-level agreement (SLA) and stores data, runs computations and queries, and provides results without alteration. However, such CSP may access data and gain information by inferring from queries and results.

A malicious adversary can manipulate data and query results, and even delete stored data, which compromises integrity and availability. Since current known CSPs are well-established companies such as Google or Amazon, it is hard to see the possibility of them behaving maliciously, as this would damage their reputation and have a negative impact on their revenues [22]. Nevertheless, the malicious adversary is taken into consideration in some settings [13], [23]. In this paper, we consider the honest but curious adversary model, i.e., the CSP or insider.

C. REFERENCE SECURITY MODEL

Security model is used to define the security guarantees of an encryption scheme. Basically, an encryption scheme is characterized by $\Pi=(\mathcal{M}, \mathcal{C}, \mathcal{K}, Enc, Dec)$, where $\mathcal{M}$ is all possible plaintexts, $\mathcal{C}$ is a set of all possible ciphertexts, and $\mathcal{K}$ a set of all possible keys belonging to a key space. Enc is a randomized algorithm where $Enc: \quad \mathcal{K} \times \mathcal{M} \longrightarrow \mathcal{C}$ and Dec is a deterministic algorithm $Dec: \quad \mathcal{K} \times \mathcal{C} \longrightarrow \mathcal{M}$. For all possible plaintexts, the following property should be satisfied [24].

$$\forall m \in \mathcal{M}, \quad \forall k \in \mathcal{K}: \quad Dec(k, Enc(k, m)) = m.$$

There are two fundamental types of encryption schemes: *symmetric-key* and *public-key* encryption. In symmetric-key encryption, the encryption and decryption keys are identical while in public-key encryption there is a key for encryption and another key for decryption. In public-key schemes, the encryption key is publicly available for encryption, but only the authorized users who have access to the decryption key can decrypt ciphertexts. We use `PPT` to denote the class of algorithms that are in probabilistic polynomial time.

To define the security model of an encryption scheme, the first step is defining a security goal. Then, it must be examined whether this goal can be achieved by an adversary. To this end, an experiment is defined between the adversary and a challenger and the probability that the adversary *wins* the experiment is studied. This experiment is called *security model*. An adversary's *advantage* is a measure of how much more successful it is at winning the experiment compared to the random guess.

For an encryption scheme Π, the following game is defined between an *adversary* $\mathcal{A}$, which tends to break the system and a *challenger*, $\mathcal{C}$, who receives as input variable b where $b \in \{0, 1\}$. First, the adversary chooses two equal-length plaintexts $m_0, m_1 \in \mathcal{M}$. The challenger picks one of two plaintexts, encrypts it and sends the result to the adversary. Then, the adversary should not be able to guess

which plaintext is encrypted by the challenger. This model is known as IND-CPA (Indistinguishability under chosen-plaintext attack), which is formalized in the following definition.

An encryption scheme, Π, is IND-CPA if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ "*wins*" the following game is negligible.

1) Challenger $\mathcal{C}$ takes a random key k
2) Adversary $\mathcal{A}$ chooses m_0, $m_1 \in \mathcal{M}$ such that $|m_0| = |m_1|$ and sends them to $\mathcal{C}$
3) Challenger $\mathcal{C}$ chooses $b \in \{0, 1\}$ uniformly at random and sends $c_b = Enc(k, m_b)$ to $\mathcal{A}$
4) $\mathcal{A}$ outputs $b' \in \{0, 1\}$, and is said to *win* if $b' = b$

In other words, if we define the **advantage** of the adversary $\mathcal{A}$ as follows.

$$Adv^{cpa}_{\mathcal{A},\Pi} = Pr[b' = b] - 1/2$$

then, the encryption scheme Π is **IND-CPA** secure if $Adv^{cpa}_{\mathcal{A},\Pi}$ is negligible, i.e., $\mathcal{A}$ cannot guess b except with probability close to $1/2$.

Note that in the above game, the adversary can issue several queries adaptively one after the other in order to simulate the ability of distinguishing multiple ciphertexts encrypted under the same key.

IND-CPA implies that the knowledge of ciphertexts provides no information about the underlying plaintexts for an adversary, i.e., ciphertexts leak no information about the plaintexts.

D. OBJECTIVES OF CLOUD-BASED DATA ANALYTICS

Secure cloud-based data analytics must target the following goals.

- **Privacy:** Data privacy must be preserved in cloud-based data analytics. Encryption is a promising solution because the adversary and the CSP can access only encrypted data, but ciphertexts may leak some information and compromise data privacy. Hence, IND-CPA is desired to achieve this in cloud-based data analytics.
- **Functionality:** Proposed solutions should efficiently support different kinds of analytical workload, i.e., exact match queries (equality checking, `GROUP BY`, `JOIN`, `DISTINCT`), range queries (inequality checking, `SORT`, `ORDER BY`), and aggregation queries (`SUM`, `AVG`, `MIN`, and `MAX`) should be supported efficiently and effectively [17]. Moreover, computations over multiple columns in a table in a database must be handled. For instance, multiplication of two columns, $C = A \times B$.
- **Efficiency:** A secure cloud-based solution should be efficient in terms of computational, storage, and communication overhead. Computational and storage overhead should be minimized for both the user and the CSP. Basically, in data outsourcing scenarios, it is considered that the user has limited storage and computational resources, thus, that overhead must be diminished as far as possible. At the CSP, the pay-per-use model of cloud computing is a good reason to minimize the total overhead of the system. Communication overhead implies the number of intermediate results that is transferred to the user for query post-processing.

III. CRYPTOGRAPHIC METHODS

There are two main cryptographic approaches that enable processing over encrypted data without decryption. We describe these cryptographic schemes, which can be implemented in practical systems for preserving data privacy in this section.

A. HOMOMORPHIC ENCRYPTION

Homomorphic Encryption (HE) allows performing arbitrary arithmetic operations over encrypted data without decryption [25]. HE provides IND-CPA security. Typically, an HE scheme is a scheme with an additional evaluation algorithm, *Eval*, to process over encrypted data [26]. The evaluation algorithm takes a public key, Pk, a function, f, two ciphertexts, c_1 and c_2 as inputs and outputs ciphertext c^* where $c^* = Eval(pk, f, c_1, c_2)$. In other words, *Eval* manipulates the encryption of two plaintexts m_1 and m_2, $c_1 = Enc(pk, m_1)$ and $c_2 = Enc(pk, m_2)$, and outputs $Enc(pk, f(m_1, m_2))$ (Figure 2).

If an HE scheme allows performing arbitrary computation, then the scheme is called Fully Homomorphic Encryption (FHE) [25]. FHE allows performing arithmetic operations $(+, -, \times, \div)$ over encrypted data without decryption. FHE is a powerful scheme and offers the highest level of security and certainly has a role to play in privacy preserving query processing [27]. FHE scheme is first introduced in [25], followed by other researches to improve the performance of the original scheme [13], [13], [28]–[34]. Even though many improvements e.g., reducing encryption key size or eliminating some complicated phases, FHE is prohibitively slow and requires so much computing power that it cannot be used in practice. As a result, building an efficient and usable FHE is still a great challenge.

Partially Homomorphic Encryption (PHE) for specific operations is efficient and can be used in practice. PHE allows either addition or multiplication over encrypted data, but not both. PHE offers the same security guarantees, IND-CPA, while being more efficient and closer to practical implementation. If a PHE scheme allows addition or multiplication, it is referred to as additive homomorphic and multiplicative homomorphic, respectively.

Paillier's scheme [35] is an example of additive homomorphic scheme and is currently the most efficient. With Paillier's scheme, multiplying the encryption of two values is equal to the encryption of the sum of the values, i.e., $Enc(k, m_1) \times Enc(k, m_2) = Enc(k, m_1 + m_2)$, where multiplication is performed modulo some public-key k [12]. Paillier's scheme is used in practical solutions [12], [21] to compute `SUM` and `AVG` aggregation queries over encrypted data without decryption.

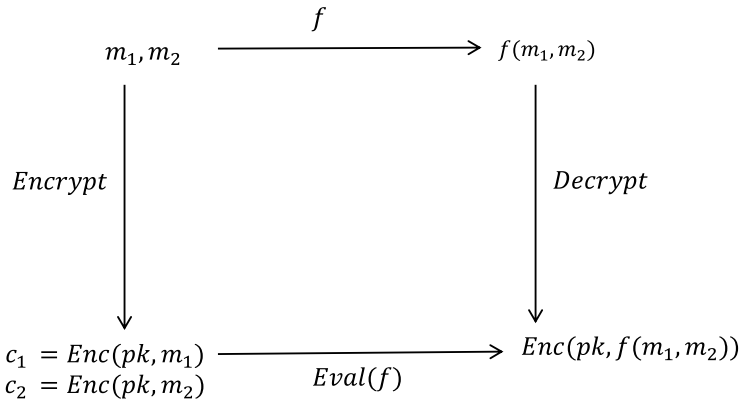


FIGURE 2. Homomorphic encryption.

B. PROPERTY-PRESERVING ENCRYPTION

Property-Preserving Encryption (PPE) preserves certain properties of plain texts on the corresponding ciphertexts, which enables the CSP to compute over encrypted data. Typically, in PPE ideal security is relaxed to provide more efficient solutions [36]. Order-preserving encryption (OPE) and deterministic encryption (DET) are examples of PPE schemes.

1) DETERMINISTIC ENCRYPTION

Deterministic (DET) encryption encrypts the same plaintext into identical ciphertexts, thus the equality property is preserved. DET allows equality checking by encrypting a plaintext into the same ciphertexts when using the same key. Thus:

$$\forall k \in \mathcal{K}, \forall m_1, m_2 \in \mathcal{M}, \quad Enc(k, m_1) = Enc(k, m_2), \quad \text{iff} \quad m_1 = m_2.$$

DET allows performing `SELECT` with equality predicates, equality `JOIN`, `GROUP BY`, `COUNT`, and `DISTINCT` queries [14]. DET is not IND-CPA secure. The following game shows that a DET encryption scheme, Π_{DET}, is not IND-CPA secure.

1) Challenger $\mathcal{C}$ chooses $b \in \{0, 1\}$ uniformly at random
2) The adversary $\mathcal{A}$ chooses $m_0, m_1 \in \mathcal{M}$ so that $m_0 = m_1 = m_*$
3) $\mathcal{C}$ sends $c_* = Enc(k, m_*)$ to $\mathcal{A}$
4) $\mathcal{A}$ chooses $m'_0 = m_*, m'_1 \neq m_*$
5) $\mathcal{C}$ sends $c = Enc(k, m'_b)$ to $\mathcal{A}$
6) $\mathcal{A}$ outputs 0 if $c = c_*$ and 1 otherwise

Hence, $Adv^{cpa}_{\mathcal{A},\Pi}$=1/2, i.e., non-negligible.

As a result, the adversary can learn data duplicates, which leads to information leakage. In order to define the security of DET schemes, the IND-CPA security game is replaced by a new security game, IND-DCPA (Indistinguishability under Distinct Chosen Plaintext Attacks), where the adversary is restricted to pick distinct plaintexts [37]. An encryption scheme, Π, is IND-DCPA if for any `PPT` adversary $\mathcal{A}$, the probability that $\mathcal{A}$ "*wins*" in the following game is negligible [38].

1) Challenger $\mathcal{C}$ takes a random key
2) Adversary $\mathcal{A}$ chooses m_0 and m_1 where m_0 and m_1 are distinct
3) Challenger $\mathcal{C}$ chooses $b \in \{0, 1\}$ uniformly at random and sends $c_b = Enc(k, m_b)$ to $\mathcal{A}$
4) $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and is said to win if $b = b'$

In the other word, considering the **advantage** of the adversary as follows

$$Adv^{dcpa}_{\mathcal{A},\Pi} = Pr[b' = b] - 1/2$$

the encryption scheme Π is **IND-DCPA** secure if $Adv^{dcpa}_{\mathcal{A},\Pi}$ is negligible.

IND-DCPA is weaker than IND-CPA, but they are equivalent when the domain of plaintexts contains unique values [39].

2) ORDER-PRESERVING ENCRYPTION

Order-Preserving Encryption (OPE) is a deterministic encryption scheme that preserves the order of plaintexts in ciphertexts [39], i.e., for any key k,

$$\forall k \in \mathcal{K}, \forall m_1, m_2 \in \mathcal{M}, \quad Enc(k, m_1) \leq Enc(k, m_2), \quad \text{iff} \quad m_1 \leq m_2.$$

OPE allows performing range queries when given encrypted constraints $Enc(k, c1)$ and $Enc(k, c2)$ corresponding to range [c1, c2]. Aggregation queries `MIN`, `MAX`, `ORDER BY`, and `SORT` can also be computed directly over encrypted data.

Since OPE is deterministic it is not IND-CPA secure. The security definition of OPE is defined by Boldyreva et al. in [39] which is called IND-OCPA (Indistinguishability under Ordered Chosen Plaintext Attacks). An IND-OCPA secure scheme hides all information about the plaintext values except the order, which is a minimum requirement for order-preserving property. An encryption scheme, Π, is IND-OCPA if for any `PPT` adversary $\mathcal{A}$, the probability that $\mathcal{A}$ "*wins*" in the following game is negligible [39].

1) Challenger $\mathcal{C}$ takes a random key
   For $i = 1, \ldots, q$
   a) Adversary $\mathcal{A}$ chooses $m_{i,0}$ and $m_{i,1}$ where $|m_{i,0}| = |m_{i,1}|$

b) Challenger $\mathcal{C}$ chooses $b \in \{0, 1\}$ uniformly at random and sends $c_b = Enc(k, m_{i,b})$ to $\mathcal{A}$ where $m_{1,0}, \ldots, m_{q,0}$ are distinct and $m_{1,1}, \ldots, m_{q,1}$ are distinct and $m_{\ell,0} < m_{j,0} \leftrightarrow m_{\ell,1} < m_{j,1}, \quad 1 \leq \ell, j \leq q$

c) $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and is said to win if $b = b'$

In the other word, considering the **advantage** of the adversary $\mathcal{A}$ as follows

$$Adv^{ocpa}_{\mathcal{A},\Pi} = Pr[b' = b] - 1/2$$

the encryption scheme is **IND-OCPA** secure if $Adv^{ocpa}_{\mathcal{A},\Pi}$ is negligible. OPE is a weaker encryption scheme than DET because in addition to leak data duplicates, it reveals the order of plaintexts.

Different OPE schemes are proposed in both the database and cryptography community. OPE is introduced in the database community by Agrawal et al. as a tool to support efficient range queries over encrypted data [40], which maps each value of the plaintext domain to one value in the ciphertext domain. This scheme bears weak privacy protection because the original data values can statistically be estimated by an adversary who has access to ciphertexts [41]. Damiani et al. propose an order-preserving indexing method, which preserves the order of plaintexts over indexes [42]. A B+tree index is built over plaintexts and is stored at the CSP in encrypted format. In order to query processing, the encrypted tree must be travelled by the CSP. However, the user should perform a sequence of queries to retrieve nodes, which induces communication overhead. The proposed scheme provides IND-OCPA security, but incurs heavy communication and computation overhead. Sobati-M. et al. introduce an order-preserving indexing method, which destroys the frequency distribution of plaintexts [36]. Whilst the proposed indexing method provides IND-OCPA security, it is efficient in terms of communication and computation overhead. Boldyreva et al. introduce a new OPE with a random mapping that preserves order [39]. The proposed scheme cannot provide IND-OCPA because it leaks at least half of the plaintext bits (i.e., more information than OPE) [43].

Popa et al. introduce the first practical IND-OCPA scheme in the cryptography community, mutable order-preserving encoding (mOPE) [43]. mOPE requires an interactive protocol for query processing. Additionally, mOPE relies on user-defined functions (UDFs) for query processing, which makes it unsuitable for cloud outsourcing. Liu et al. introduce an OPE scheme that randomly splits the original plaintext domain into successive intervals with different lengths [41]. Then, an extended ciphertext domain is selected and split into the same number of intervals. Finally, nonlinear mapping functions map the original plaintexts into ciphertexts in the extended domain. The proposed method partially destroys the distribution of original data and reveals some information about underlying values, which breaks IND-OCPA security.

Table 1 shows the comparison between existing OPE schemes. Among all OPE approaches, only Boldyreva's scheme is implemented in some practical solutions.

TABLE 1. Comparison of OPE schemes.

Scheme	Interactive	IND-OCPA
Agrawal	✘	✘
Damiani	✔	✔
Sobati	✘	✔
Boldyreva	✘	✘
Popa	✔	✔
Liu	✘	✘

IV. ENCRYPTION-BASED PRACTICAL SOLUTIONS

Building a secure cloud-based analytics system has been discussed briefly in the literature with limited practical contribution. Some solutions have nonetheless been proposed. The most important challenge of secure solutions is weak security guarantees or heavy overhead in the systems we review in this section, to the best of our knowledge.

For each practical solution, we explain the main idea, the architecture, how data are organized at the CSP, and query processing. Then, we analyze the advantages and the deficiencies of each solution in terms of security and performance.

A. BUCKETIZATION

Bucketization is introduced as a privacy-preserving method in [10], [11] that allows partial execution of a query at the CSP with the help of indexes (Figure 3). Bucketization supports queries over encrypted data without decryption. Queries are evaluated in an approximate manner, thus the returned results may contain some false-positives. The final result is found by decrypting the data and executing query post-processing at the users.

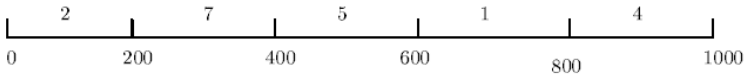
Bucketization divides data into buckets and provides explicit labels for each bucket [44]. The domain of plaintexts is partitioned into a set of non-overlapping buckets (subsets), with the same size (or maybe different size). A label is defined for each bucket and may or may not preserve the order of values in the original domain. Then, bucket labels are stored along with encrypted values at the CSP. These labels allow equality, range (if preserving order) and join queries at the CSP without decryption. Bucketization-based indexing usually returns false-positives in the result of a query. Thus, query post-processing is needed at the users to filter out false-positives [45].

Any relation R in a database is encrypted in tuple-level and auxiliary indexes, i.e., buckets's labels, are used by the CSP for query processing. Thus, the relation $R(A_1, A_2, ..., A_n)$ is stored at the CSP as: $R^S(etuple, A^S_1, A^S_2, ..., A^S_n)$, where the attribute *etuple* corresponds to an encrypted tuple, and each A^S_i corresponds to the index for the attribute A_i. An example is shown in Figure 3.

The DO must maintain metadata such as the label of bucket for transforming queries to the appropriate representation on the CSP and performing query post-processing.

eid	ename	salary	addr	did
23	Tom	70K	Maple	40
860	Mary	60K	Main	80
320	John	50K	River	50
875	Jerry	55K	Hopewell	110

(a)



(b)

$etuple$	eid^S	$ename^S$	$salary^S$	$addr^S$	did^S
1100110011110010...	2	19	81	18	2
1000000000011101...	4	31	59	41	4
1111101000010001...	7	7	7	22	2
1010101010111110...	4	71	49	22	4

(c)

FIGURE 3. (a) An example of a relation R, (b) partition function used for attribute *eid*, and (c) the encryption relation R^S stored at the CSP [10].

In Bucketization architecture, queries are transformed by a *query translator* and a *query executor*, which both reside with the DO. For query processing, the AU poses the query to the query translator.

Each query is transformed into the server-side and user-side sub-queries, Q^s and Q^c, respectively (Figure 4). Q^s is executed by the CSP over encrypted data using corresponding indexes ($A_i^S s$).

The result of Q^s is sent back to the query executer, which decrypts the result of Q^s and executes Q^c to filter false-positives and retrieve the final result (post-processing).

Query execution in bucketization operates as follows.

- *Step1.* The AU sends a query Q to the query translator. The query translator rewrites Q into Q^s and Q^c. To this end, the query translator uses some metadata (e.g.,the buckets boundaries) and replaces the plaintext values in the query with bucket boundaries. Then, the query translator sends Q^s to the CSP.
- *Step2.* The CSP executes Q^s and returns all encrypted records back, whose bucket numbers satisfy the query constrains. The encrypted results are sent to the query executor.
- *Step3.* The query executor decrypts the results and executes Q^c to eliminate false-positives and sends the final results to the AU.

Bucketization provides efficient query processing while keeping the information disclosure to a minimum. Furthermore, query evaluation is often much simpler than cryptographic schemes [46].

However, there is a trade-off between security and efficiency. The smaller number of buckets leaks less information and increases security, but induces more false-positives in query results and more computation overhead at the AU.

When the labels are order-preserving, IND-CPA security is not guaranteed. Note that here the encrypted data is in the form of $(ciphertext)||(bucketlabel)$, e.g., in Figure 3 the ciphertexts of attribute *eid* are in the form of $etuple||eid^S$. Additionally, bucketization-based indexing reduces data granularity. All values in a row are encrypted together, which means that all encrypted rows must be shipped back to the AU inducing communication overhead.

B. CRYPTDB

CryptDB is a secure DataBase Management System (DBMS) developed at MIT [12] with both academic [18], [20], [21], [47]–[53] and industrial [54]–[58] impacts [14], e.g., Google [54] and SAP [55] produced their own CryptDB-inspired solutions. CryptDB aims at providing data privacy guarantees in the face of a compromised server and a honest but curious CSP by data encryption. CryptDB uses a set of encryption schemes based on the queries issued by the AU [14] and adopts different kinds of encryption schemes, i.e., PPE and PHE, which are dynamically adjusted depending on the queries [26]. Encryption in CryptDB is like onion layers that store multiple ciphertexts within each other (Figure 5).

For instance, given two encryption schemes $\texttt{Enc}^1$ and $\texttt{Enc}^2$, the encryption of a value m is defined as:

$$c = Enc_1(k_1, (Enc_2(k_2, m))).$$

The outermost onion layers provide the highest level of security, IND-CPA security, whereas the inner layers, provide more functionality and less security guarantees, e.g., IND-DCPA for DET layer.

Each value in a relation is encrypted independently. Numeric values are maintained in three different onions,

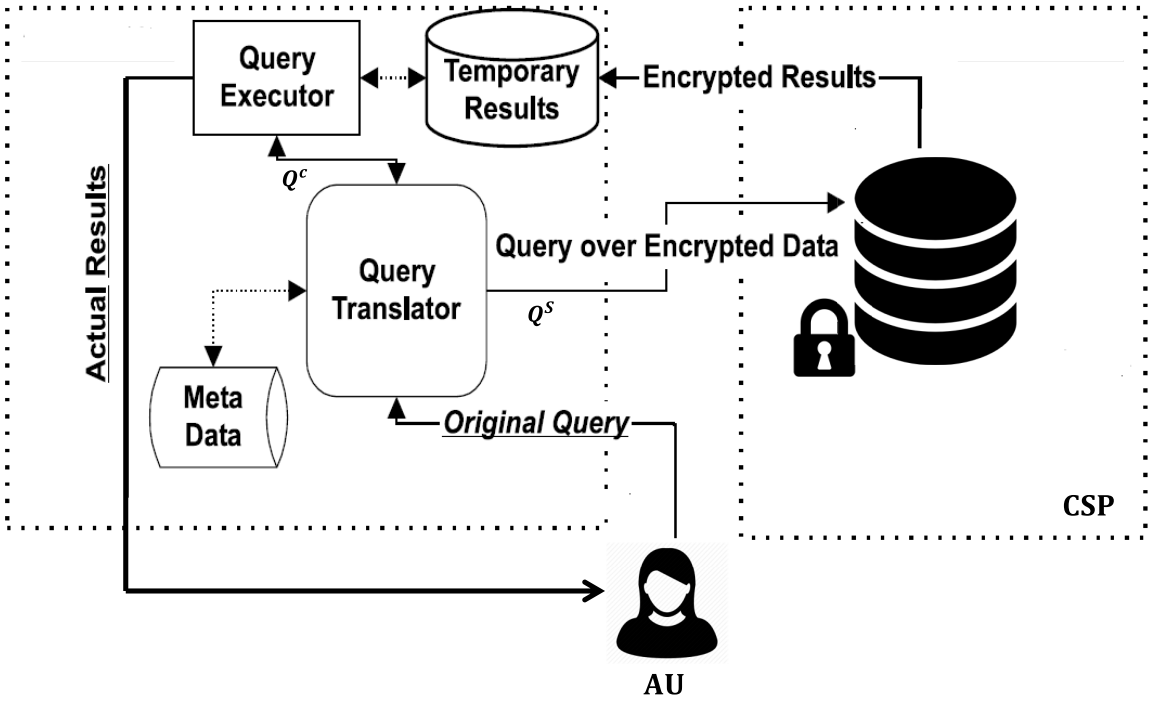


FIGURE 4. Bucketization query workload [10].

OnionEq, OnionOrd, and OnionAdd, which are used for equality checking, range queries, and aggregations `SUM` and `AVG`, respectively. In other words, for each value CryptDB stores three encrypted values at the CSP. At the CSP, query processing is computed using UDFs.

In CryptDB architecture, a proxy server intercepts communications between the user and the CSP and applies en/decryption of queries and results (Figure 6).

To create an encrypted database from a database, the proxy server generates a master secret key and uses it to encrypt each relation in the database [59]. The proxy server stores the master key and the scheme of the database and uses them for query rewriting. When a query is issued, the proxy dynamically peels off onion layers down to a layer corresponding to the given computation. Onion layers are adjusted by sending the related key and updating the columns. Then, the proxy server anonymizes each table and column name and encrypts each constant in the issued query with an encryption scheme corresponding to the requested operation.

Query execution in CryptDB operates as follows.

- *Step1*. The AU sends a query Q to the proxy. The proxy rewrites Q into Q^s operating at the CSP. To this end, the proxy encrypts all constants in Q adopting the encryption scheme that best suits the operation to be computed [26].
- *Step2*. The proxy checks if the CSP should be given keys to remove some encryption layers. In this case, the proxy issues an `UPDATE` query that removes specific layers of encryption, and then sends Q^s to the CSP.

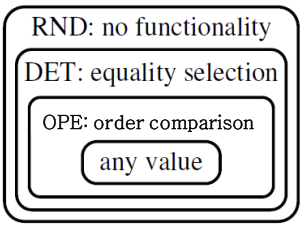


FIGURE 5. An example of onion encryption layers in CryptDB [12].

- *Step3*. The CSP computes Q^s and returns the encrypted results to the proxy.
- *Step4*. The proxy decrypts the encrypted results and sends the final results to the AU.

For instance, in order to evaluate a range query, each value of an attribute is encrypted using an OPE scheme (first encryption layer). Then, the resulting OPE ciphertexts are encrypted with a randomized encryption (RND) scheme (second encryption layer). A randomized encryption scheme encrypts the same values into different ciphertexts using the same key and allows no computation whilst providing the highest level of security, IND-CPA. When a range query is issued, the proxy server sends the decryption key for the second encryption layer to the CSP. The CSP then decrypts the randomized ciphertexts and gets access to order-preserving ciphertexts. Hence, the CSP learns the order of values and evaluates the range query [60] and sends the encrypted values to the proxy server.

CryptDB supports standard SQL queries over encrypted data and needs no change to existing applications. Basically, applications can transparently run on top of CryptDB.

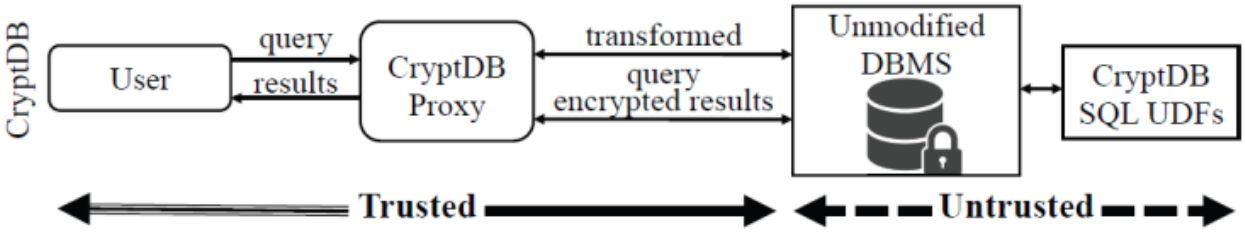


FIGURE 6. CryptDB architecture [12].

However, CryptDB is targeted for transactional workloads. It is not feasible to run analytical workloads over CryptDB [61]. As a result, CryptDB supports only 2 queries out of 22 queries from the TPC-H benchmark.

Some computations are not supported, for instance, computing both summation and comparison on encrypted values. Thus, to execute the following query.

Q1 :

```
SELECT SUM(price) AS total
       FROM orders
    GROUP BY order_id
    HAVING total >100
```

CryptDB cannot check if `total >100` at the CSP, because `total` is encrypted with a PHE scheme, i.e., the Paillier scheme, which does not support order comparison [21].

Complex queries over multiple columns in a relation, for instance, the following query

Q2 :

SELECT SUM($t_1.B * t_2.A$) FROM T as t_1 WHERE $t_1.A = t_2.B$.

cannot be evaluated because multiplication of two encrypted values is not supported in CryptDB.

Moreover, onion layers become an overhead as well as peeling off a layer, especially in the case of big tables. CryptDB performs almost all queries at the CSP with a relatively small overhead in terms of query throughput [12]. However, the throughput is worst for queries that use PHE, such as summations because, in order to compute summations, CryptDB implements modular multiplications at the CSP. Improving the performance of aggregations is important because aggregation is common in data analytics.

Although onions offer multiple levels of security but reveal different information about the data [22]. It is easy to see that security decreases over time when the outer layers are removed. Because adjustable encryption architecture in CryptDB is unidirectional, i.e., once a column is decrypted to a weaker scheme like DET, it never returns to a higher encryption level [62].

C. MONOMI

MONOMI extends CryptDB's functionality to support analytical queries [21]. In contrast to CryptDB that focuses on transactional workloads, MONOMI mainly targets analytical workloads [61].

Instead of using the trusted proxy server, MONOMI splits query processing between the CSP and the user. Using user/server query splitting, MONOMI executes as much of the query as is practical over encrypted data at the CSP and executes remaining computations by the user, which decrypts data and processes queries further [63]. To this end, MONOMI introduces an optimized designer and a planner. The designer chooses an appropriate database design based on the target workload. Like CryptDB, MONOMI implements different cryptographic schemes, but unlike CryptDB, there is no onion of encryption. To choose a set of encryption schemes, MONOMI uses an optimizing designer similar to physical designers used by other databases, which takes the kind of computations that are likely to appear in future queries, e.g., `SUM` for attribute Salary (Figure 7). In other words, the designer can be considered similar to automated index selection and materialized view selection tools.

MONOMI designer is invoked when a database is loaded by the DO. The DO must also provide a query workload $Q_1, Q_2, \ldots, Q_n$ to represent the operations that the AU will perform over data. Given the user's inputs, MONOMI designer provides a physical design consisting a set of encryption schemes for each relation. For each query Q_i $i = 1, \ldots, n$, the designer determines a set of required encryption schemes and invokes the planner to determine how to best execute Q_i given the encryption schemes. MONOMI planner determines different plans by determining for each plan what parts of Q_i would be executed by the AU and what parts at the CSP. Then, a cost model is used by the planner to estimate the cost of each plan (e.g., execution times). The planner chooses the fastest plan for Q_i and denotes the corresponding subset of encryption schemes.

Once the best plan is determined for each query, the designer takes the union of the required encryption schemes and uses them for physical design. Hence, a plaintext value is encrypted using different cryptographic schemes. Further, the planner selects the query execution path for each query given a particular physical design.

In order to execute queries that cannot be computed at the CSP alone, MONOMI partitions query execution across the CSP, which has access only to encrypted data, and the AU, who has access to the decryption keys.

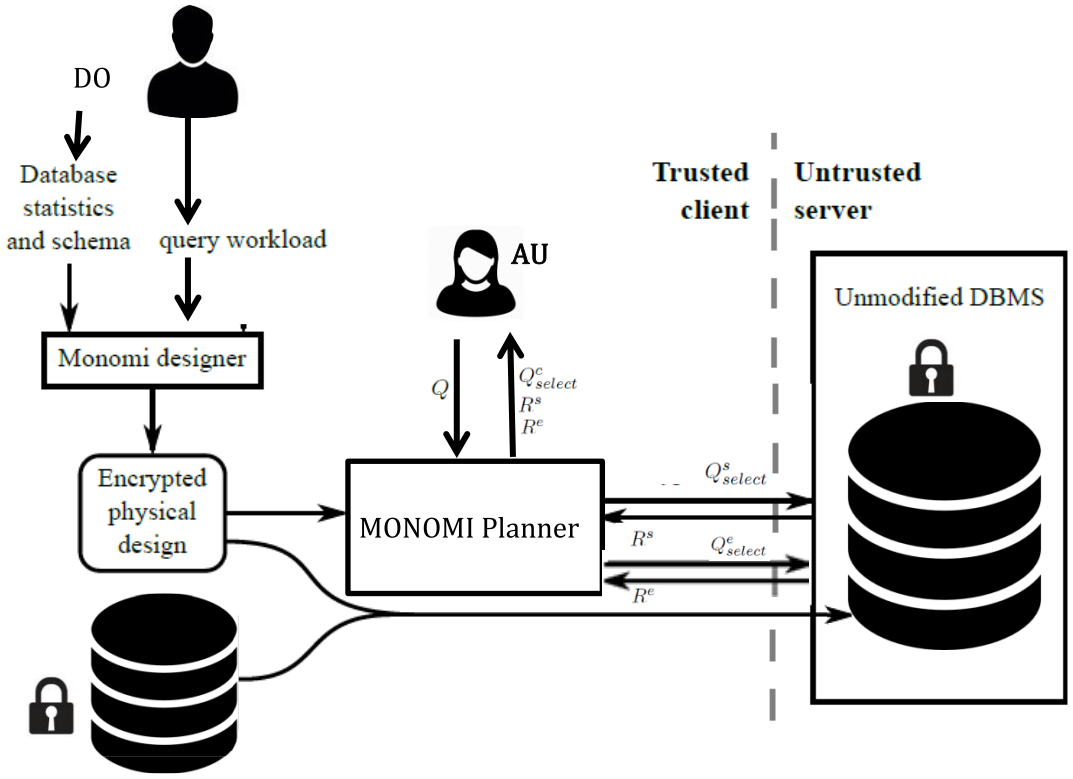


FIGURE 7. MONOMI architecture [21].

However, query splitting technique cannot be optimal in all situations and depends on data. To choose an optimal splitting, MONOMI models the cost of query execution for each query plan as the sum of execution time at the CSP, transfer time, and post-processing time at the AU's side. Then, the lowest-cost plan is chosen.

Query execution in MONOMI consists the following steps.

- *Step1*. The AU sends a query Q to MONOMI planner. The planner computes different plans $P^Q_1, P^Q_2, \ldots, P^Q_\ell$, where $P^Q_i \quad i = 1, \ldots, \ell$ consists of three sub queries Q^s_i, Q^e_i, and Q^c_i. The sub-query Q^s_i is performed by the CSP over encrypted data, Q^e_i asks for retrieving some encrypted data $R^e_i = \{e_1, e_2, \ldots, e_n\}$ from the CSP, and Q^c_i is performed by the AU over retrieved encrypted values, R^e_i, after decryption.
  For each plan $P^Q_i \quad i = 1, \ldots, \ell$, the planner computes a cost $C^Q_i = t_1 + t_2 + t_3$ where t_1, t_2, and t_3 are execution times at the CSP, transferring time, and post-processing time by the AU, respectively. Then, a plan with the lowest-cost is selected by the planner. We show the selected plan as P^Q_{select}. The planner sends Q^s_{select} and Q^e_{select} to the CSP.
- *Step2*. The CSP computes Q^s_{select} over encrypted data and returns encrypted results, R^s, to the planner. The CSP also retrieves encrypted values, R^e, corresponding to Q^e_{select} and sends them to the planner, too.
- *Step3*. The planner sends the results R^s and R^e to the AU. The planner also sends Q^c_{select} to the AU.
- *Step4*. The AU decrypts encrypted values in R^e and executes Q^c_{select} over unencrypted values. The AU also decrypts R^s to obtain the final results.

Like CryptDB, MONOMI implements PHE [35] for computing aggregation queries `SUM` and `AVG`, which is computationally intensive and induces a large ciphertext size. However, large ciphertext size imposes storage cost and affects query processing. Additionally, PHE requires modular multiplications, which is computationally expensive. To overcome these drawbacks, MONOMI introduces some optimization techniques. To avoid performing separate modular multiplications, MONOMI concatenates several plaintext values into a single larger plaintext that will be encrypted as a single value. In this way, MONOMI reduces storage overhead and simultaneously decreases the number of modular multiplications.

Query splitting enables executing more queries with optimal costs. However, it requires that queries are declared ahead of time by the user, which is not possible for all scenarios, especially for ad-hoc analytical workloads. Moreover, it is essential to cut down the bandwidth required to transfer intermediate results, computation and storage resources for the AU for query processing [21], because resources usage at the AU must be minimum for maintaining the benefits of outsourcing.

Optimization techniques used by MONOMI improve performance, but also induce some limitations. Packing values

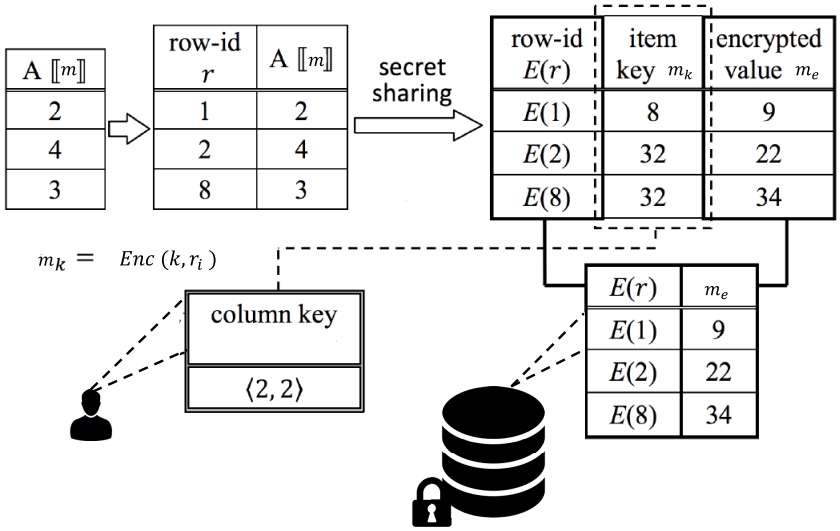


FIGURE 8. Encryption of attribute *A* in SDB [64].

into a single value reduces storage overhead and also amortizes computation overhead, but makes impossible partial operation or partial query processing. For example, when packing multiple values in a column into a single plaintext, a query that updates only a part of the column cannot be executed.

D. SDB

SecureDB (SDB) is a secure query processing system, which allows a wide range of complex queries to be computed by the CSP [64]. SDB introduces a new encryption scheme based on so called *secret sharing*. In secret sharing, each plaintext value is split into several shares that are stored at n users' [65]. One single user has no means to reconstruct the secret, but a subset of $k \leq n$ users can reconstruct the secret [66].

Each plaintext value, m_i, is split into two shares, one kept at the DO, which is referred as *item key*, m_k, and another at the CSP, which is regarded as ciphertext, m_e. In order to reduce data storage at the DO's side, item keys is considered to be the same for all values. In order to encrypt the plaintext values in an attribute, first a random secret key, k, is defined. Then, for each row of attribute a random row-id, r_i, is defined, too. Then, the row-ids are encrypted by the secret key k, $m_k = Enc(k, r_i)$ and m_i is encrypted using encrypted row-id, m_k, which is $m_e = Enc'(m_k, m)$. The values of row-ids are encrypted and stored along with the ciphertexts at the CSP (Figure 8).

The proposed scheme simulates FHE, which leads to a wide range of queries supported by the CSP. The proposed encryption scheme supports *data interoperability*, i.e., the output of an operator is used as the input of another operator. SDB computes complex analytical queries over multiple columns in a table using data interoperability. For instance, computing "$A+B > 1000k$" is possible in SDB, but the same computation is not supported by the other secure systems like CryptDB.

SDB architecture is similar to CryptDB's in which a proxy server is set between the DO/AU and the CSP.

Query execution in SDB operates as follows.

- *Step1*. The AU sends a query, Q, to the proxy server. The proxy rewrites Q into sub queries Q^s operating at the CSP and Q^c which is computed at the proxy server.
- *Step2*. The proxy server executes Q^c and produces new secret keys.
- *Step3*. The proxy server sends the new secret keys and Q^s to the CSP. Note that the new keys generated by the proxy are needed for computing the sub-query Q^s at the CSP.
- *Step4*. The CSP computes Q^s and returns the encrypted results to the proxy.
- *Step5*. The proxy decrypts the encrypted results and sends the final results to the AU.

SDB sends an `UPDATE` query to the CSP for some operations. For instance, consider two columns A and B, which are encrypted with two different keys, a_k and a_b. To compute $C = A + B$, the proxy server generates a new key c_k using a_k and a_b and sends c_k to the CPS. The CSP executes the `UPDATE` query to encrypt A and B with the new key, c_k, and gets new columns A' and B', and then adds the encrypted values of A' and B'.

SDB requires modular multiplication for decryption, which induces heavy computation overhead at the proxy server. Computations at the CSP also consist of modular exponential, which is expensive and affects query processing.

While the proposed scheme supports a wide range of queries, there are still some queries that are not supported, e.g., keyword search over encrypted strings.

Security guarantees in SDB is the highest, i.e., IND-CPA security. Yet, IND-CPA reveals no information about the underlying plaintexts, which makes impossible using optimization techniques, e.g., B+tree indexing. As a result,

query processing needs scanning the whole database, leading to poor performance. Like CryptDB and MONOMI, SDB strongly relies on UDFs at the CSP for query processing, which makes it unsuitable for cloud-based scenarios because the DO has no permission to create UDFs, e.g., small enterprises deploy their secure web-based system using the rented database [41].

E. TRUSTEDDB

TrustedDB is an outsourced database prototype that implements tamper-proof trusted hardware for secure query processing in the cloud [15]. A secure co-processor (SCPU) such as the IBM 4764/5 [67] is deployed at the CSP's side to compute query processing over encrypted data. The SCPU provides a secure computation environment. However, SCPUs are constrained in both computation ability and memory capacity. Hence, data are classified as being either private, sensitive or insensitive public [68]. The former are encrypted and operations are performed inside the SCPU, and the latter are stored unencrypted at the CSP. All sensitive data are encrypted by the DO before uploading to the CSP. The entire database is stored outside the SCPU. Data are encrypted using a symmetric encryption scheme such as AES. Some randomness is also added to ensure IND-CPA security. However, such randomized encryption schemes allow no computation over encrypted data. Sensitive encrypted data that need to be accessed by the SCPU for query processing are pulled in by the SCPU.

An issued query is rewritten by the AU into two sub-queries, which are executed by the CSP and the SCPU [15]. The SCPU in TrustedDB consists of a *Request Handler*, a *Query Parser*, and a *Query Dispatcher* to handle query rewriting (Figure 10).

Query execution in TrustedDB operates as follows.

- *Step1*. The AU encrypts a query using the public key used by the SCPU and sends the encrypted query to the CSP.
- *Step2*. The CSP forwards the encrypted query to the SCPU. The Request Handler (RH) receives the encrypted query.
- *Step3*. The RH decrypts the query and sends the unencrypted query to the Query Parser (QP).
- *Step4*. The QP rewrites the query into three sub-queries, Q^s, Q^e, and Q^{TC}, and sends them to the Query Dispatcher (QD).
- *Step5*. The QD sends Q^s and Q^e to the CSP.
- *Step6*. The CSP executes Q^e and retrieves requested encrypted data, R^e, and sends them to the QD. The CSP also executes Q^s over unencrypted data and sends the results back again to the QD.
- *Step7*. The QD decrypts data in R^e and sends decrypted values along with Q^{TC} to the DB engine residing inside the CSPU. After executing Q^{TC}, the final results are re-encrypted by the QD and sent to the RH. Finally, the RH sends the encrypted results to the AU.

Note that the execution of private queries, i.e., Q^{TC}, depends on the output of public queries, i.e., Q^s, and vice-e-versa.

Using the SCPU and a randomized encryption scheme, TrustedDB simulates FHE, i.e., the security guarantees of TrustedDB is IND-CPA. However, leaving some data unencrypted at the CSP may compromise the privacy of encrypted sensitive data by linking between them. Moreover, sending encrypted data to the SCPU incurs communication overhead, which affects query processing and results in poor performance. Sending the final results to the QD, which encrypts the results imposes computation overhead.

F. CIPHERBASE

CipherBase is another solution based on tamper-proof trusted hardware to preserve the confidentiality of sensitive data [20]. A FPGA (Field Programmable Gate Array) is set at the CSP as a trusted component, TC, which computes some computations over sensitive data on behalf of the CSP. Computations are decomposed between the trusted TC and the CSP. Like TrustedDB, the sensitivity of data must be defined by the DO on the scheme definition. Insensitive public data are stored unencrypted at the CSP. Highly sensitive data, e.g., *Patient.ID*, are encrypted using a strong encryption scheme, i.e., an IND-CPA secure scheme. Less sensitive data, e.g., *Patient.Age* are encrypted by a weaker encryption scheme, e.g., a PPE scheme.

Highly sensitive data are shipped to the TC, which are decrypted and processed. In fact, CipherBase simulates FHE by integrating non-homomorphic encryption schemes (e.g., AES in CBC mode) with trusted hardware's to compute any operation.

Query processing in CipherBase consists of several round trips between the TC and the CSP. A *query planner*, which resides at the AU's, is responsible for query re-writing Fig. 10.

The query planner rewrites an issued query into three sub-queries. Among them, two sub-queries are executed by the CSP and the third sub-query is sent to the TC. Query execution in CipherBase operates as follows.

- *Step1*. The query planner rewrites an issued query, Q, into three sub-queries, Q^s, Q^e, and Q^{TC} and sends them to the CSP.
- *Step2*. The CSP executes Q^e and retrieves encrypted tuples, R^e, and sends them along with Q^{TC} to the TC. The CSP also computes Q^s over unencrypted data.
- *Step3*. The TC decrypts all tuples in R^e and executes Q^{TC} over decrypted data.
- *Step4*. The TC encrypts the results and ships them back to the CSP.
- *Step5*. The CSP sends all results consisting encrypted and unencrypted results to the AU.

Security guarantees in CipherBase is specified by the DO. The highest level of security achieves when all data are highly sensitive. However, security comes at a cost of poor performance. Specifying some data as less sensitive and insensitive

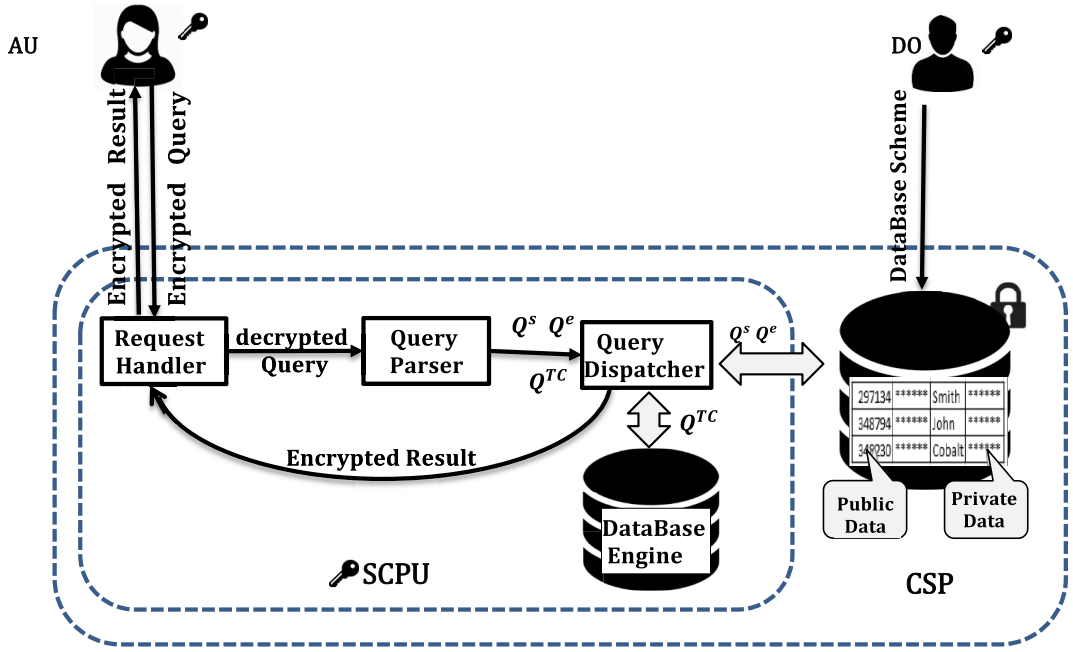


FIGURE 9. TrustedDB architecture and query processing workload.

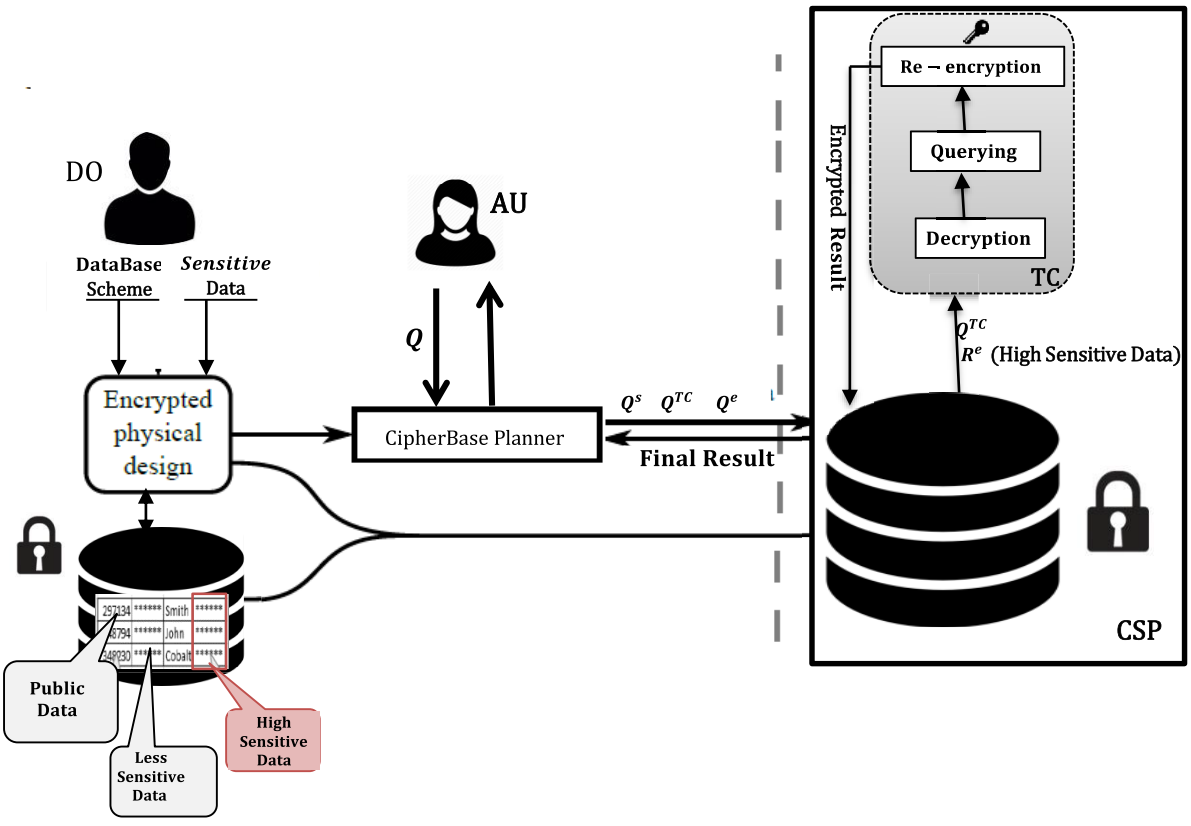


FIGURE 10. CipherBase architecture and query processing workload.

by the DO degrades security guarantees and increases performance. In this case, CipherBase provides the same level of security as CryptDB.

Query processing in CipherBase involves shipping encrypted data from the CSP to the TC, decrypting, query processing, and re-encrypting data in TC, and shipping results

back to the CSP. Moving encrypted data between the CSP and the TC incurs communication overhead. Moreover, decryption and re-encryption are also costly in terms of computation. Whilst some optimization techniques are considered to minimize round trips, but the overhead of decryption/re-encryption is still a bottleneck.

G. COMPARISON OF EXISTING SOLUTIONS

In this section, we compare the existing solutions presented in Section IV regarding to overhead, security, trusted component, and query support features. Table 2 summarizes the features of all solutions, which we discuss below.

1) COMMUNICATION OVERHEAD

Communication overhead is defined as the number of encrypted records sent as the result of an issued query. Consider a query Q, the encrypted results that satisfy the query is shown as $|Q(e)|$. Bucketization induces a communication overhead of $|Q(e)|+|Q(f)|$ where $|Q(f)|$ indicates the number of false-positives. Since MONOMI planner requests retrieving some encrypted values, R^s, the communication overhead of MONOMI is $|Q(e)| + |Q(R^s)|$.

Like MONOMI, TrustedDB and CipherBase retrieve some encrypted values, R^s, and send them to the TC. Hence, the communication overhead of TrustedDB and CipherBase is $|Q(R^s)|$, which is imposed at the CSP's side when data are moved to the TC.

2) COMPUTATION OVERHEAD

Computation overhead at the user's side is defined as the time of decryption, t_e. Bucketization requires also auxiliary time, t_f, to eliminate false positives after decryption. MONOMI induces an extra time, t_d, for decrypting encrypted values in R^s and also t_{post} for query processing over decrypted values. Hence, MONOMI has the largest computation overhead at the user's side.

At the CSP, computation overhead consists of the required time for query processing over encrypted values, t_{query}. CryptDB and SDB sends also an `UPDATE` query, which induces $t_{query} + t_{\texttt{UPDATE}}$ computation overhead at the CSP. MONOMI also requires $t_{retrieve}$ for retrieving encrypted values. Nevertheless, the computation overhead of MONOMI is less than CryptDB and SDB because the `UPDATE` query consists of the re-encryption of encrypted values and contains expensive modular operations such as modular multiplication. Computation overhead for TrustedDB consists of t_p for querying public unencrypted data, $t_{retrieve}$ for retrieving encrypted private values, t_{dec} for decrypting private encrypted data, and t_{TC} for querying data after decryption inside the SCPU. CipherBase needs also t_{query} for querying less sensitive data at the CSP. However, comparing to TrustedDB, CipherBase incurs less computation overhead because CipherBase needs less data decryption because some computations are computed over less sensitive data without decryption.

3) SECURITY

Security guarantees depend directly on the cryptographic schemes which are used in a solution. Bucketization uses PHE, but cannot provide IND-CPA because the bucket's labels reveal some information about the encrypted values. Bucketization introduces a trade-off between security and efficiency; for example, when the labels are order-preserving, query processing is more efficient, but at the cost of a lower level of security. In fact, the security level of order-preserving labeling is the same as that of PPE [6].

On the other hand, having a small number of buckets increases security, but triggers more false-positives in query results, leading to poor performance and less efficiency. PHE is implemented in CryptDB and MONOMI to provide IND-CPA security; however, security is relaxed to improve the efficiency of query processing [66]. CryptDB and MONOMI implement PPE to support more queries, which discloses IND-CPA security. IND-CPA is guaranteed when no query is executed and the outermost layer of encryption, randomized encryption, has been kept. Once the randomized layer is removed, however, security is degraded to the level of PPE.

The proposed encryption scheme in SDB provides IND-CPA security. However, IND-CPA is no longer guaranteed when a query is executed. After an updating query, the values of a column are transformed in such a way that the query can be executed by the CSP. For instance, in order to execute a `GROUP BY` for a column, after updating, the encryption values are equal for the same plaintexts, and so equality leaks, which degrades the level of security to the level of PPE.

Implementing randomized encryption schemes along with the SCPU in TrustedDB guarantees IND-CPA, so TrustedDB offers the same level of security as FHE, but greater efficiency. At the same time, leaving less sensitive data unencrypted at the CSP raises concerns about linking attacks. In linking attacks, the attacker tries to find some information about sensitive data using public unencrypted data, which may compromise data privacy [86]. More importantly, security is guaranteed as long as the SCPU is not in danger [64]. Similarly, the TC in CipherBase simulates FHE by executing computations inside the TC, and security is guaranteed as long as the TC is not compromised [96]. Yet the level of security required depends on the sensitivity of the data, i.e. highly sensitive data should be encrypted with a strong encryption scheme so that no information will be revealed. Even though CipherBase provides a reasonable security level for highly sensitive data, using PPE schemes for less sensitive data degrades security guarantees.

The vulnerability of PPE to inference attacks is studied in [1]. The authors describe an inference attack directly over encrypted data. Even if a non-deterministic encryption scheme is used to encrypt data, such an attack can be executed successfully. The results convincingly illustrate the trade-offs between security, performance and functionality for query processing in a TC.

4) TRUSTED COMPONENT (TC)

The TC is set at the user's side (the AU and the DO's side) in Bucketization and MONOMI. CryptDB and SDB rely on TC-PS as the TC. Considering the DO and the AU as a TC eliminates the need of extra components. However, the TC must be installed and maintained on each AU and DO separately. Setting a proxy server adds a new point of attack and induces computational and storage cost, but eliminates the need of installing the TC for each DO and AU. The TC is configured at the CSP (TC-TPH in Figure 1) in TrustedDB and CipherBase. Whilst installing the TC at the CSP minimizes computational burdens for the AU, it needs an agreement with the CSP, which is not always feasible. Moreover, using tamper-proof hardware is still expensive and it is not affordable for small/medium businesses. More importantly, installing the TC at the CSP protects data privacy against attackers that compromise the CSP, but not insiders and the CSP who has access to the TC.

5) QUERY SUPPORT

Equality checking, inequality, and aggregation queries are supported in all systems. All those queries are executed on only one column in a table, but SDB handles operations between separated columns, e.g., addition between two columns A and B, $A + B$.

Regarding to standard benchmarks, SDB is the only system that supports all TPC-H queries. Bucketization, CryptDB, and MONOMI handle only 2, 4, and 19 out of 22 TPC-H queries, respectively. TrustedDB supports only 4 non-nested queries from TPC-H because nested queries are not supported by TrustedDB. CipherBase is designed for OLTP workloads; hence query processing for TPC-H workloads is not discussed.

V. LIMITATIONS AND OPEN ISSUES

A. PRIVACY VS. FUNCTIONALITY

Despite the highest level of security provided by FHE, current FHE schemes are inefficient for practical data analytics due to computation overhead [59]. Contrary, PHE schemes are more efficient and closer to practical solutions. However, PHE schemes provide less functionality due to supporting limited operations only. PPE schemes allow operating over encrypted data in the same way as they would operate on plaintext. PPE are crucial for practical secure solutions because of their functionality. However, such PPE schemes provide lower security guarantees, e.g., IND-DCPA security for DET encryption, leading to the leakage of a non-trivial amount of information, which makes such schemes vulnerable to some attacks (e.g., frequency analysis attacks). Recent works [1], [2], [59], [69], [70] apply and develop different attacks using the leakage of PPE schemes. For instance, Naveed et al. demonstrate that a large fraction of the records from the DET encrypted columns in a database can be decrypted when the adversary possesses statistical information about the plaintexts [59]. Hence, secure systems such as CryptDB and MONOMI sacrifice data privacy for enhancing functionality, i.e., there is a trade-off between a higher level of functionality and less data privacy.

B. PRIVACY VS. EFFICIENCY

Processing over encrypted data is computationally more expensive than original plaintext [71]. The first reason is data expansion. For instance, a 32-bit plaintext is expanded to 256-bits of ciphertext using classic AES+CBC (Cipher-Block Chaining) encryption [18] and 1,024 or more bits long using PHE. Such enormous data expansion induces both storage and computation overhead. The second reason is the nature of operations. In particular, operations at the CSP should not involve any expensive modular arithmetic operation like modular multiplication or exponentiation [72]. Whilst FHE and PHE offer the highest security guarantees, but necessitate modular operation for query processing. Security guarantees are relaxed in PPE to provide more efficiency.

Setting a trusted server like a trusted proxy server in CryptDB induces extra cost, which does not fit cloud data outsourcing. Moreover, since the trusted server has access to the encryption keys and plaintext information, it becomes an appealing target for attackers.

Splitting queries in [10], [21] to be executed partly at the CSP and partly by the AU incurs computation overhead at the end user. In data outsourcing the goal is fully outsourcing computations because the resources for the AU are limited. Moreover, transferring intermediate results to the user induces communication overhead. Hence, it is essential to cut down the bandwidth required to transfer intermediate results to the AU and reduce computation overhead at the AU's side.

Tamper-proof hardware used in TrustedDB and CipherBase is significantly constrained in both computation ability and memory capacity, hence setting such components at the CSP faces major efficiency challenges. A trade-off should be defined between more efficient untrusted main CPU computation and less efficient trusted computations inside the tamper-proof component [15]. Moreover, leaving some data unencrypted at the CSP for the sake of memory capacity limitation may disclose the privacy of encrypted sensitive data.

C. ACCESS PATTERN PRIVACY

Data protection methods, discussed before, guarantee the confidentiality and privacy of the data stored at the CSP. Another security issue in outsourcing scenarios arises when querying data whilst preserving the privacy of access patterns. In fact, by observing enough query results the adversary could infer about data and data privacy could be compromised by correlating prior information with frequently queried data [72]. Private access patterns have recently raised the attention of researchers [45]. New cryptographic schemes are needed that allow the CSP to send items in response a query without knowing which item is being sent. *Private Information Retrieval* (PIR) [73]–[76], theoretically enables accessing data items while preventing the CSP to learn anything

TABLE 2. Features of practical solutions.

		Computation		Communication		Cryptographic Schemes						Queries			
		Server	User	Server	User	FHE	PHE	PPE	TC	Security IND-CPA	TPC-H	R	E	A	M-C
Approaches	Bucketization	t_{query}	$t_e + t_f$	–	$\|Q(e)\| + \|Q(f)\|$	✘	✔	✘	TC-DO TC-AU	✘	2	✔	✔	✔	✘
	CryptDB	t_{query}+t_{UPDATE}	t_e	–	$\|Q(e)\|$	✘	✔	✔	TC-PS	✘	4	✔	✔	✔	✘
	MONOMI	t_{query}+$t_{retrieve}$	$t_e + t_d + t_{post}$	–	$\|Q(e)\| + \|Q(R^s)\|$	✘	✔	✔	TC-DO TC-AU	✘	19	✔	✔	✔	✘
	SDB	t_{query}+t_{UPDATE}	t_e	–	$\|Q(e)\|$	✔	✘	✘	TC-PS	✔	All	✔	✔	✔	✔
	TrustedDB	t_p+$t_{retrieve}$//+t_{dec}+t_{TC}	t_e	$\|Q(R^s)\|$	$\|Q(e)\|$	✔	✘	✘	TC-TPH	✔	4	✔	✔	✔	✘
	CipherBase	t_p+t_{query}+$t_{retrieve}$ + t_{dec}+t_{TC}	t_e	$\|Q(R^s)\|$	$\|Q(e)\|$	✘	✔	✔	TC-TPH	✘	–	✔	✔	✔	✘

R: Range queries, E: Equality checking, A: Aggregation, M-C: Multi Columns

about query access patterns [77]. The most of current PIR solutions aim at very strong security bounds and remain unsuitable for practical purposes [78]. Deployment of existing PIR protocols would have been orders of magnitude more time consuming than transferring the entire database to the user [79].

Oblivious RAM (ORAM) [80] provides access pattern privacy, too [79]. ORAM allows reading and writing to memory without revealing access pattern to the CSP. ORAM continuously shuffles and re-encrypts data as they are being accessed, thereby completely hiding access pattern [81].

While the idea of relying on ORAM and PIR to enable access pattern privacy is promising, but a key challenge is the efficiency of such schemes.

Another solution could be sending frequently fake queries without taking care with the results [82], [83]. The goal is to mislead the adversary of making valid inference about correlation between hot frequently queried data and other data. Submitting such queries requires more investigation on the way of generating that looks realistic.

D. DATA INTEGRITY

While a passive model is by far the most widely assumed adversary model in the literature, in some scenarios the active model should be considered. Thus, there is also the need to address an active adversary model. In an active model, the adversary has malicious intension and may change or modify the data, results of queries or even in some cases interrupt or cause denial of service. Such threats are aimed at data integrity and availability. Protecting against such an adversary needs more effort by the user to ensure the correctness of data and query results. The results must be demonstrably authentic to ensure that the data has not been tampered with (*integrity*). The proof of completeness must be carried by the results that allow the user to verify that the CSP has not omitted any valid tuples that match the queries predicate [84] (*completeness*). Such proof assures that the query is executed with completeness over their entire target data set [72]. For instance, when executing a `JOIN` query, the user should be able to verify that the CSP returned all matching values.

Authentication and integrity checking along encryption become important in such scenarios. Typically, signature or MAC (Message Authentication Code) is used to allow the user to check the integrity of returned items [23], [85]. However, existing works introduce mechanisms for efficient integrity and authentication checking only for simple queries and they are limited to the queries of some specific kind.

E. HIGH DIMENSIONAL DATA

Cloud-based EHRs (Electronic Health Record systems) are another case of cloud-based data analytics. In EHRs, a patient's sensitive data (e.g. data extracted from disease images) is stored at the CSP. The patient's data is always stored in the form of high-dimensional vectors [91]. The objective of cloud-based EHRs is to find approximate *k Nearest Neighbors (KNN)* in a privacy-preserving manner, so approximate kNN queries are executed over vectors without any leakage of information about the underlying vectors [90]. Differential Privacy techniques [93], [94] add a great amount of noise to query results, leading to low accuracy and making query results useless. However, encryption cannot be implemented for EHRs due to the heavy computation overhead for high-dimensional vectors [92]. Existing solutions focus on low-dimensional datasets and are unable to handle high-dimensional data [91].

VI. CONCLUSION

In this paper, we discuss data security issues which emerge when organizations outsource both data and data analytics to Cloud Service Providers. We review the security mechanisms which can be used for the deployment of secure cloud-based data analytics services. We focus particularly on cryptographic schemes and practical systems which enable the execution of queries over encrypted data without decryption. We highlight the benefits and drawbacks of existing solutions in a cloud computing context and suggest practical solutions.

Building practical secure systems is a challenging task because there is a trade-off between privacy and functionality/efficiency. Using FHE in practical solutions enables computing arbitrary operations, i.e., combining high functionality with the highest level of security, which is promising for cloud-based data analytics. However, the level of efficiency

of FHE is still a major problem. PHE schemes on the other hand are more efficient and therefore closer to being practical solutions, but they support partial computations only and cannot provide a completely practical solution with all the desired functionality. In PPE schemes privacy is relaxed to provide greater efficiency, and whilst they provide the same functionality as computing over unencrypted data, their poor level of security is a great challenge. Implementing tamper-proof hardware for secure query processing simulates FHE with greater efficiency, although this is affected by the limited memory capacity of such trusted hardware.

Cryptography cannot simultaneously provide the highest levels of security, efficiency and functionality. It is thus essential to specify clearly the objectives of any deployment of cloud-based data analytics and adopt cryptographic schemes which are tailored to those objectives. In future work, we plan to analyze the efficiency of various practical solutions by carrying out different tests with various datasets.

REFERENCES

[1] P. Antonopoulos *et al.* (2018). "Pushing the limits of encrypted databases with secure hardware." [Online]. Available: https://arxiv.org/abs/1809.02631

[2] F. Pallas and M. Grambow, "Three tales of disillusion: Benchmarking property preserving encryption schemes," in *Trust, Privacy and Security in Digital Business*, S. Furnell, H. Mouratidis, and G. Pernul, Eds. Cham, Switzerland: Springer, 2018, pp. 39–54.

[3] D. Naous, J. Schwarz, and C. Legner, "Analytics as a service: Cloud computing and the transformation of business analytics business models and ecosystems," in *Proc. 25th Eur. Conf. Inf. Syst. (ECIS)*, May 2017, pp. 125–136.

[4] S. Schulte, C. Janiesch, S. Venugopal, I. Weber, and P. Hoenisch, "Elastic business process management: State of the art and open challenges for BPM in the cloud," *Future Gener. Comput. Syst.*, vol. 46, pp. 36–50, May 2015.

[5] E. Damiani, S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Key management for multi-user encrypted databases," in *Proc. ACM Workshop Storage Secur. Survivability*, Fairfax, VA, USA, Nov. 2005, pp. 74–83.

[6] E. Shi, J. Bethencourt, T.-H. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range Query over encrypted data," in *Proc. IEEE SP*, May 2007, pp. 350–364.

[7] M. Golfarelli, S. Rizzi, and I. Cella, "Beyond data warehousing: What's next in business intelligence?" in *Proc. DOLAP*, Washington, DC, USA, Nov. 2004, pp. 1–6.

[8] G. Amanatidis, A. Boldyreva, and A. O. , "Neill, "Provably-secure schemes for basic query support in outsourced databases," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, Redondo Beach, CA, USA, 2007, pp. 14–30.

[9] J. L. D., Jr., and C. V. Ravishankar, "Security limitations of using secret sharing for data outsourcing," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, Paris, France, Jul. 2012, pp. 145–160.

[10] H. Hacigümüs, B. R. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. ACM SIGMOD*, Madison, Wisconsin, Jun. 2002, pp. 216–227.

[11] H. Hacigümüş, B. R. Iyer, and S. Mehrotra, "Efficient execution of aggregation queries over encrypted relational databases," in *Proc. DASFAA*, Jeju Island, Korea, 2004, pp. 125–136.

[12] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted Query processing," in *Proc. 13rd ACM Symp. Oper. Syst. Princ.*, Oct. 2011, pp. 85–100.

[13] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Jun. 2010, pp. 24–43.

[14] R. A. Popa, "Building practical systems that compute on encrypted data," Ph.D. dissertation, MIT, Cambridge, MA, USA, 2014.

[15] S. Bajaj and R. Sion, "TrustedDB: A trusted hardware-based database with privacy and data confidentiality," in *Proc. SIGMOD*, Athens, Greece, Jun. 2011, pp. 205–216.

[16] V. Attasena, N. Harbi, and J. Darmont. (2014). "FVSS: A new secure and cost-efficient scheme for cloud data warehouses." [Online]. Available: https://arxiv.org/abs/1412.3538

[17] M. A. Hadavi and R. Jalili, "Secure data outsourcing based on threshold secret sharing; Towards a more practical solution," in *Proc. VLDB PhD Workshop*, 2010, pp. 54–59.

[18] A. Arasu, K. Eguro, M. Joglekar, R. Kaushik, D. Kossmann, and R. Ramamurthy, "Transaction processing on confidential data using cipherbase," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Apr. 2015, pp. 435–446.

[19] T. Ge and S. B. Zdonik, "Answering aggregation queries in a secure system model," in *Proc. VLDB*, Sep. 2007, pp. 519–530.

[20] A. Arasu *et al.*, "Orthogonal security with cipherbase," in *Proc. 6th Biennial Conf. Innov. Data Syst. Res. (CIDR)*, Asilomar, CA, USA, Jan. 2013.

[21] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," *Proc. VLDB Endowment*, vol. 6, no. 5, pp. 289–300, 2013.

[22] B. K. Samanthula, W. Jiang, and E. Bertino, "Privacy-preserving complex Query evaluation over semantically secure encrypted data," in *Proc. ESORICS*, Wroclaw, Poland, Sep. 2014, pp. 400–418.

[23] M. A. Hadavi, M. Noferesti, R. Jalili, and E. Damiani, "Database as a service: Towards a unified solution for security requirements," in *Proc. IEEE COMPSAC*, Izmir, Turkey, Jul. 2012, pp. 415–420.

[24] C. Yildizli, T. B. Pedersen, Y. Saygin, E. Savas, and A. Levi, "Distributed privacy preserving clustering via homomorphic secret sharing and its application to (vertically) partitioned spatio-temporal data," *Int. J. Data Warehousing Mining*, vol. 7, no. 1, pp. 46–66, Jan. 2011.

[25] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2009.

[26] F. H. Liu, "Computation over encrypted data," in *Cloud Computing Security: Foundation Challenges*. Boca Raton, FL, USA: CRC Press, 2016, pp. 305–320.

[27] M. D. Ryan, "Cloud computing security: The scientific challenge, and a survey of solutions," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2263–2268, Sep. 2013.

[28] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Proc. CRYPTO*, Santa Barbara, CA, USA, 2011, pp. 505–524.

[29] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proc. CRYPTO*, Santa Barbara, CA, USA, vol. 2013, pp. 75–92.

[30] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu, "Private database queries using somewhat homomorphic encryption," in *Proc. ACNS*, Banff, AB, Canada, Jun. 2013, pp. 102–118.

[31] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Proc. PKC*, Paris, France, May 2010, pp. 420–443.

[32] C. Gentry and S. Halevi, "A working implementation of fully homomorphic encryption," in *Proc. EUROCRYPT*, Nice, French, May 2010, pp. 1–5.

[33] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys," in *Proc. CRYPTO*, Santa Barbara, CA, USA, Aug. 2011, pp. 487–504.

[34] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, *(Leveled) Fully Homomorphic Encryption Without Bootstrapping*. Cambridge, U.K.: Cambridge Univ. Press, Jan. 2012, pp. 309–325.

[35] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Mar. 1999, pp. 223–238.

[36] S. Sobati-Moghadam, G. Gavin, and J. Darmont, "A secure order-preserving indexing scheme for outsourced data," in *Proc. IEEE Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2016, pp. 297–303.

[37] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proc. CRYPTO*, Santa Barbara, CA, USA, Aug. 2007, pp. 535–552.

[38] M. Bellare, M. Fischlin, and A. O'Neill, and T. Ristenpart, "Deterministic encryption: Definitional equivalences and constructions without random oracles," in *Proc. CRYPTO*, Santa Barbara, CA, USA, Aug. 2008, pp. 360–378.

[39] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *Advances in Cryptology—EUROCRYPT*. 2012, p. 624.

[40] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order-preserving encryption for numeric data," in *Proc. ACM SIGMOD*, Paris, France, Jun. 2004, pp. 563–574.

[41] Z. Liu, X. Chen, J. Yang, C. Jia, and I. You, "New order preserving encryption model for outsourced databases in cloud environments," *J. Netw. Comput. Appl.*, vol. 59, pp. 198–207, 2016.

[42] E. Damiani, S. D. C. di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Balancing confidentiality and efficiency in untrusted relational DBMSs," in *Proc. ACM CCS*, Washington, DC, USA, Oct. 2003, pp. 93–102.

[43] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 463–477.

[44] E. Mykletun and G. Tsudik, "Aggregation queries in the database-as-a-service model," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2006, pp. 89–103.

[45] P. Samarati and S. D. C. di Vimercati, "Data protection in outsourcing scenarios: Issues and directions," in *Proc. ACM ASIACCS*, Beijing, China, Apr. 2010, pp. 1–14.

[46] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu, "Secure multidimensional range queries over outsourced data," *VLDB J.*, vol. 21, no. 3, pp. 333–358, Jun. 2012.

[47] A. Arasu, K. Eguro, R. Kaushik, D. Kossmann, R. Ramamurthy, and R. Venkatesan, "A secure coprocessor for database applications," in *Proc. 23rd Int. Conf. Field Program. Logic Appl.*, Sep. 2013, pp. 1–8.

[48] S. D. Tetali, M. Lesani, R. Majumdar, and T. D. Millstein, "MrCrypt: static analysis for secure cloud computations," *ACM SIGPLAN*, vol. 48, no. 10, pp. 271–286, Oct. 2013.

[49] J. Kepner *et al.*, "Computing on masked data: a high performance method for improving big data veracity," in *Proc. IEEE HPEC*, Waltham, MA, USA, Sep. 2014, pp. 1–6.

[50] J. J. Stephen, S. Savvides, R. Seidel, and P. Eugster, "Practical confidentiality preserving big data analysis," in *Proc. 6th USENIX Workshop Hot Topics Cloud Comput.*, Philadelphia, PA, USA, Jun. 2014, pp. 786–796.

[51] H. Shafagh, "Toward computing over encrypted data in IoT systems," *ACM Crossroads*, vol. 22, no. 2, pp. 48–52, Aug. 2015. doi: 10.1145/2845157.

[52] H. Shafagh, A. Hithnawi, A. Droescher, S. Duquennoy, and W. Hu, "Poster: Towards encrypted query processing for the internet of things," in *Proc. MobiCom*, Sep. 2015, pp. 251–253.

[53] H. Shafagh, L. Burkhalter, and A. Hithnawi, "Talos a platform for processing encrypted IoT data: Demo abstract," in *Proc. ACM SenSys*, Stanford, CA, USA, Nov. 2016, pp. 308–309.

[54] Google. (2015). *Encrypted Big Query*. [Online]. Available: https://github.com/google/encrypted-bigquery-client

[55] P. Grofig *et al.*, "Privacy by encrypted databases," in *Proc. APF*, Athens, Greece, May 2014, pp. 56–69.

[56] (2018). *Always Encrypted*. [Online]. Available: https://msdn.microsoft.com/enus/library/mt163865(v=sql.130).aspx

[57] Dotissi SRL. *CryptonorDB*. Accessed: Jul. 2018. [Online]. Available: http://www.cryptonordb.com/

[58] (2018). *Lincoln Laboratory*. [Online]. Available: http://www.ll.mit.edu/index.html

[59] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proc. SIGSAC*, Oct. 2015, pp. 644–655.

[60] J. Köhler, "Tunable security for deployable data outsourcing," Ph.D. dissertation, Dept. Comput. Sci., 2015.

[61] E. Saleh, "Processing over encrypted data: Between theory and practice," in *Proc. 8th PhD. Retreat HPI Res. School Service-Oriented Syst. Eng.*, Aug. 2015, pp. 63–73.

[62] F. Kerschbaum *et al.*, "Adjustably encrypted in-memory column-store," in *Proc. ACM SIGSAC*, Berlin, Germany, Nov. 2013, pp. 1325–1328.

[63] Q. Cai, J. Lin, F. Li, and Q. Wang, "SEDB: building secure database services for sensitive data," in *Proc. ICICS*, Hong Kong, China, Dec. 2014, pp. 16–30.

[64] Z. He *et al.*, "SDB: A secure query processing system with data interoperability," *VLDB Endowment*, vol. 8, no. 12, pp. 1876–1879, Aug. 2015.

[65] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[66] S. S. Moghadam, J. Darmont, and G. Gavin, "Enforcing privacy in cloud databases," in *Big Data Analytics and Knowledge Discovery*, vol. 10440, L. Bellatreche and S. Chakravarthy, Eds. New York, NY, USA: Springer, 2017, pp. 53–73.

[67] IBM 4765 PCIe. (2018). *Cryptographic Coprocessor*. [Online]. Available: http://www-03.ibm.com/security/cryptocards/ pciecc/overview.shtml

[68] S. Bajaj and R. Sion, "Trusteddb: A trusted hardware based outsourced database engine," *VLDB*, vol. 4, no. 12, pp. 1359–1362, Feb. 2011.

[69] F. B. Durak, T. M. DuBuisson, and D. Cash, "What else is revealed by order-revealing encryption?" in *Proc. ACM SIGSAC*, Oct. 2016, pp. 1155–1166.

[70] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill, "Generic attacks on secure outsourced databases," in *Proc. ACM SIGSAC*, Vienna, Austria, Oct. 2016, pp. 1329–1340.

[71] R. L. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 82–105, Jan. 2013.

[72] R. Sion, "Towards secure data outsourcing,"in *Handbook Database Security*, M. Gertz and S. Jajodia, Eds. Boston, MA, USA: Springer, 2008.

[73] W. Lueks and I. Goldberg, "Sublinear scaling for multi-client private information retrieval," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, Jul. 2015, pp. 168–186.

[74] C. Cachin, S. Micali, and M. Stadler, "Computationally private information retrieval with polylogarithmic communication," in *Proc. EUROCRYPT*, 1999, pp. 402–414.

[75] Y. Chang, "Single database private information retrieval with logarithmic communication," in *Proc. Australas. Conf. Inf. Secur. Privacy*, Jul. 2004, pp. 50–61.

[76] E. Kushilevitz and R. Ostrovsky, "One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval," in *Proc. EUROCRYPT*, Bruges, Belgium, 2000, pp. 104–121.

[77] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, 1998.

[78] E. Mykletun, M. Narasimha, and G. Tsudik, "Signature bouquets: Immutability for aggregated/condensed signatures," in *Proc. ESORICS*, Sep. 2004, pp. 160–176.

[79] P. Williams and R. Sion, "Usable PIR," in *Proc. NDSS*, San Diego, CA, USA, Feb. 2008, pp. 1–6.

[80] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, no. 3, pp. 431–473, 1996.

[81] E. Stefanov *et al.*, "Path ORAM: An extremely simple oblivious RAM protocol," in *Proc. ACM SIGSAC*, Berlin, Germany, Aug. 2013, pp. 299–310.

[82] C. Mavroforakis, N. Chenette, and A. O'Neill, G. Kollios, and R. Canetti, "Modular order-preserving encryption, revisited," in *Proc. ACM SIGMOD*, May 2015, pp. 763–777.

[83] H. Pang, X. Xiao, and J. Shen, "Obfuscating the topical intention in enterprise text search," in *Proc. IEEE ICDE*, Washington, DC, USA, Apr. 2012, pp. 1168–1179.

[84] M. Narasimha and G. Tsudik, "Authentication of outsourced databases using signature aggregation and chaining," in *Proc. ACM CIKM*, Bremen, Germany, Oct. 2005, pp. 235–236.

[85] V. Attasena, N. Harbi, and J. Darmont, "A novel multi-secret sharing approach for secure data warehousing and on-line analysis processing in the cloud," *Int. J. Data Warehousing Mining*, vol. 11, no. 2, pp. 22–43, 2015.

[86] A. Harmanci and M. Gerstein, "Quantification of private information leakage from phenotype-genotype data: Linking attacks," *Nature Methods*, vol. 13, no. 3, pp. 251–256, 2016.

[87] C. Guo, X. Chen, Y. Jie, F. Zhangjie, M. Li, and B. Feng, "Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption," *IEEE Trans. Services Comput.*, to be published.
[88] Z. Shen, J. Shu, and W. Xue, "Preferred search over encrypted data," *Frontiers Comput. Sci.*, vol. 12, no. 3, pp. 593–607, 2018.
[89] C. Guo, R. Zhuang, Y. Jie, Y. Ren, T. Wu, and K. R. Choo, "Fine-grained database field search using attribute-based encryption for e-healthcare clouds," *J. Med. Syst.*, vol. 40, no. 11, pp. 235–243, Feb. 2016.
[90] W. Wang, L. Chen, and Q. Zhang, "Outsourcing high-dimensional healthcare data to cloud with personalized privacy preservation," *Comput. Netw.*, vol. 88, pp. 136–148, Sep. 2015.
[91] W. C. Xue, H. Li, Y. Peng, J. Cui, and Y. Shi, "Secure*k*Nearest neighbors Query for high-dimensional vectors in outsourced environments," *IEEE Trans. Big Data*, vol. 4, no. 4, pp. 586–599, Dec. 2018.
[92] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013.
[93] C. Xu, J. Ren, Y. Zhang, Z. Qin, and K. Ren, "DPPro: Differentially private high-dimensional data release via random projection," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 12, pp. 3081–3093, Dec. 2017.
[94] S. Su, P. Tang, X. Cheng, R. Chen, and Z. Wu, "Differentially private multi-party high-dimensional data publishing," in *Proc. IEEE ICDE*, May 2016, pp. 205–216.
[95] D. Naous, J. Schwarz, and C. Legner, "Analytics as a service: Cloud computing and the transformation of business analytics business models and ecosystems," in *Proc. 25th Eur. Conf. Inf. Syst. (ECIS)*, Jun. 2017, pp. 487–501.
[96] D. Kossmann, "Confidentiality à la carte with cipherbase," in *Datenbanksysteme für Business, Technologie Und Web*. Bonn, Germany: Gesellschaft für Informatik, 2017, pp. 23–24.

Somayeh Sobati Moghadam received the master's degree from INSA de Lyon, France, and the Ph.D. degree from Lyon2 University, France, in 2016. She is currently an Assistant Professor with the Electrical and Computer Engineering Faculty, Hakim Sabzevari University, Iran. She has authored several journal and conference papers. Her current researchs include privacy, security issues in cloud computing, decision support systems, data mining, the IoT, and business information systems.

AMJAD FAYOUMI received the Ph.D. degree in information systems area from Loughborough University, for his thesis focusing on the use of enterprise modeling and simulation for analyzing and designing business strategy and operation. He is currently a Lecturer in information systems (IS) with the Management Science Department, Lancaster University Management School. He is researching in the information systems area focusing on enterprise modeling and simulation. His recent research interest includes architecting digital enterprises with a particular focus on the use of emergent intelligent systems in organizations and societies. He is currently a member of a number of professional associations and he has taken roles in reviewing and editing research papers for several international conferences and journals.