Integrity Authentication for SQL Query Evaluation on Outsourced Databases: A Survey

Bo Zhang, Boxiang Dong, and Wendy Hui Wang

Abstract—Spurred by the development of cloud computing, there has been considerable recent interest in the Database-as-a-Service (DaaS) paradigm. Users lacking in expertise or computational resources can outsource their data and database management needs to a third-party service provider. Outsourcing, however, raises an important issue of result integrity: how can the client verify with lightweight overhead that the query results returned by the service provider are correct (i.e., the same as the results of query execution locally)? This survey focuses on categorizing and reviewing the progress on the current approaches for result integrity of SQL query evaluation in the DaaS model. The survey also includes some potential future research directions for result integrity verification of the outsourced computations.

Index Terms—Database-as-a-Service, SQL query, result integrity verification

1 INTRODUCTION

The amount spent by corporations, non-profit organizations, and government agencies in implementing and supporting data management and analytics is considerable. Globally, 3,500 enterprises spend, on average, $664,000 annually on data management [1]. Due to the fast growth of data volumes, the scale of data management systems is increasingly crossing the petabyte barrier [2], [3], [4]. Unfortunately, the ability of organizations to support effective and efficient data management typically lags behind their ability to collect and store the data. The solution of in-house data analytics software is not satisfactory, as it either may not be able to address users’ specific data analysis needs or fails to deliver data management services that are easily deployable. On the other hand, hiring in-house data management professionals is not affordable by small and medium-sized organizations that have limited financial budget.

Explosive development of the Internet and advances in networking technology have fueled a new computing paradigm called Database-as-a-Service (DaaS) [5]. In the DaaS paradigm, the clients who own large volumes of data but lack resources to do data management themselves outsource their data as well as the data analysis to a third-party database service provider. The service provider offers adequate hardware, software, and network resources to host the clients’ databases, and provides technical supports of data management services, including data access, query processing, and dealing with updates. Several industrial organizations such as Amazon, Google and Microsoft are providing cloud-based database services in various forms. For example, Amazon Web Services (AWS) provides computation capacity and data storage via Amazon Elastic Compute Cloud (EC2) [6] and Simple Storage Service (S3) [7]. Google provides Cloud SQL [8], a fully-managed database service for relational PostgreSQL and MySQL databases. Microsoft provides cloud database services [9] on Azure cloud [10]. By using these services, the clients can exploit the benefit of mass storage, accelerated processing capacity, and sophisticated data management and analytics at a low cost.

Outsourcing database management to a computationally powerful service provider enables to achieve sophisticated analysis on large volumes of data in a cost effective way. With this architecture, however, the client no longer has direct control over the outsourced data and computations. Without any security guarantee from the service provider, clients have little faith in the security of the received outsourced services. One of the important security issues of DaaS paradigm is the result integrity of the outsourced computations. The outsourced data or the computations may be corrupted due to malware or security break-ins [11], [12]. Or a malicious insider (e.g., a disgruntled employee) could modify the program and/or the query results. Given the fact that many data analytics applications (e.g., fraud detection and business intelligence) are mission critical, it is important to provide efficient and practical methods to enable the client to verify whether the DaaS service provider returned correct results of the outsourced database services.

Intuitively, the general-purpose protocols for verifiable computations [11], [12] can support the verification of any arbitrary query. However, due to the generality, these approaches can incur excessive proof construction overhead at the server side [13]. In the last two decades, a large variety of efficient authentication methods have been proposed.
for specific types of SQL queries. A brief survey of a subset of these methods was presented in [14]. In this survey article, we will give a comprehensive overview of the existing approaches for result integrity authentication of the DaaS paradigm. By doing the survey, we hope to provide a useful resource for both database and security communities.

The rest of the survey is organized as follows. Section 2 gives an overview of the integrity verification goals and a categorization of the existing authentication methods. Section 3 presents the preliminaries. Sections 4 and 5 discuss the existing authentication techniques that return deterministic and probabilistic integrity guarantee respectively. Section 6 discusses the complexity of the authentication methods and some practical issues. Section 7 presents the solutions that consider integrity with additional security features (e.g., access control and privacy). Finally, Section 8 concludes the survey and discusses the possible research directions for the future work.

2 OVERVIEW

In this section, we provide an overview of the Database-as-a-Service (DaaS) paradigm, the verification goals, and the categorization of the existing authentication methods for outsourced SQL query evaluation (see Table 1 and Fig. 1).

2.1 Database-as-a-Service (DaaS) Paradigm

The concept of the database-as-a-service (DaaS) paradigm is first defined in [5]. A typical database-as-a-service (DaaS) paradigm consists of three entities: (1) the data owner who possesses a large collection of records $D$. Due to the lack of resources, the data owner outsources $D$ to a service provider; (2) the service provider (SP) who provides storage and data management services for the clients; and (3) the client who sends queries to SP. SP executes the queries on the outsourced data, and returns the query results to the client. The client may have limited computational power (e.g., she may use mobile devices for verification). The data owner and the client may be the same entity.

In this survey, we mainly focus on relational databases and SQL queries, including range queries, join, and aggregate SQL queries. The authentication methods studied in this survey cover the following types of SQL queries:

- **Range** queries, including single-dimensional range queries with selection condition on the index attribute only [15], [16], [17], [18], [19] and on any single attribute [20], and multi-dimensional range queries [13], [14], [21], [22], [23], [24], [25], [26], [27], [28];
- **Join** queries, including equi-join [19], [25], [29], [30], [31], [32], [33], theta-join [14], [21], [27], [28], [31], and multi-join [14], [26], [27], [34];
- **Aggregate** queries, including SUM [14], [23], [25], [26], [27], [31], MAX/MIN [23], [25], [27], COUNT [23], [25], [27], and AVG [23], [25], [27];
- **Mixed** queries, including aggregate queries of multi-dimensional range [23], equi-join on single-dimensional range queries [31], join and aggregate on multi-dimensional range queries [14], [25], [26], and general nested queries [27].

2.2 Verification Goal

One of the serious concerns of the DaaS paradigm is that SP may return incorrect query evaluation results, due to various reasons such as server errors or external compromise [26]. The goal of result integrity authentication is to enable the client to verify that for any query $Q$, if the results $R$ returned by SP are the same as by executing $Q$ on the dataset $D$ locally, i.e., if $R = Q(D)$. There are three different authentication goals: soundness, completeness, and freshness. Next, we discuss the three authentication goals in details.

- **Soundness.** All the records in $R$ must satisfy $Q$. Intuitively, the soundness of $R$ can be measured by the precision of $R$. 

| Paper | Query Type | Verification Goal | Authentication Method |
|-------|------------|-------------------|-----------------------|
| MHT [22] | Range | $S$ | Deterministic Approach |
| ASM, AIM, ASM [74] | Range | $S$ | Probabilistic Approach |
| VB-tree [55], XB-Tree [58], PMD [49] | Range | $C$ | |
| MB-tree, EMB-tree [42] | Range | $F$ | |
| APS-Tree, AAB-Tree [43] | Range | $F$ | |
| FIAT [53] | Join | $S$ | |
| Aggregate signature [50] | Join | $F$ | |
| DSAC [51], Iterative hash [54] | Join | $C$ | |
| ECC-signature [56] | Join | $F$ | |
| VKD-Tree [18] | Join | $F$ | |
| iDAX [79] | Join | $F$ | |
| AMR [57] | Join | $F$ | |
| IntellegiDB [78] | Join | $F$ | |
| vSQL [77] | Join | $F$ | |
| CorrectDB [9] | Join | $F$ | |
| Query sampling [65] | Join | $F$ | |
| Fake records [21], [23] | Join | $F$ | |
| Markers/twins/saltis [72] | Join | $F$ | |

**TABLE 1**

Summary of the Existing Authentication Techniques for Outsourced SQL Query Evaluation ($S$: Soundness; $C$: Completeness; $F$: Freshness; $TP$: Interactive-Proof; $Upd.$: Update; $Agg.$: Aggregate Queries; $Sign.$: Signature; $Th$: Trusted Hardware)
The query is sound if precision is 1.

\[
\text{Precision}(R) = \frac{|R \cap Q(D)|}{|R|}
\]

The query is complete if recall is 1.

\[
\text{Recall}(R) = \frac{|R \cap Q(D)|}{|Q(D)|}
\]

The query is complete if recall is 1.

- **Completeness.** \( R \) must include all records that satisfy \( Q \).
  The completeness of \( R \) can be measured by recall of \( R \).

- **Freshness.** When there are updates on the outsourced dataset \( D \), the query \( Q \) must be executed against the latest data records, rather than out-of-date versions.
  The returned results \( R \) only include the records from the latest dataset.

### 2.3 A Categorization of Integrity Authentication Techniques

In this survey, we categorize the existing result integrity authentication approaches into two types, based on the verification guarantee that these approaches can return:

- The deterministic approaches that assure that for any probabilistic polynomial-time adversary, the probability to produce an incorrect query result and pass verification is negligible according to a given security parameter.
- The probabilistic approaches that catch any incorrect query result with a certain probability that does not exceed a user-specified threshold (not necessarily close to 1).

Next, we briefly summarize the key ideas of these two types of authentication approaches.

#### 2.3.1 Deterministic Authentication

Consider the query \( Q \) and its results \( R \), all deterministic authentication solutions rely on proofs of the query results to return a deterministic integrity guarantee. Intuitively, besides the query results \( R \), SP constructs a short proof of \( R \), and sends both the proof and \( R \) to the client. The proof sometimes is referred to as the verification object (VO). The client utilizes the proof to verify that \( R \) satisfies certain requirements (e.g., soundness, completeness, and freshness). At a high level, most of the proof-based authentication methods follow the similar procedure below. Before sending the dataset \( D \) to SP, the data owner computes some auxiliary information \( A \) of \( D \), which is known as an authentication data structure (ADS). The data owner sends both \( D \) and \( A \) to SP, while keeping \( A \) locally. She may distribute certain information, for example, the root digest of the ADS, to the clients. After receiving the query \( Q \) from a client, SP executes \( Q \) on \( D \), and constructs a proof of the query results \( R \), by using both \( R \) and the auxiliary information \( A \) obtained from the data owner. SP sends both the query results and the proof to the client. The client verifies result integrity by using both the proof and certain information that the data owner shared with her. Fig. 2 illustrates the framework of the proof-based authentication methods.

Based on how the proof is constructed, we categorize the proof-based methods into five types, namely tree-based solutions, signature-based solutions, accumulation value based solutions, interactive proof based solutions, and trusted hardware based solutions.

- **Tree-based methods.** The ADS normally takes the tree format. As part of query execution, SP traverses the ADS tree and gathers the information of respective nodes to form the proof. From the query results and the proof, the client re-constructs the traversal path used in query execution and verifies that it is indeed authentic.
- **Signature-based methods.** The data owner signs individual records in a chain fashion, and uploads the dataset with signatures to SP. During query execution, SP gathers the signatures of the records in the query results, and compresses these signatures into an aggregated one, and use the aggregate signature as the proof. The client authenticates the results by utilizing the signatures of the records in the results and the aggregate signature in the proof.
- **Interactive proof based methods.** These methods extend the information-theoretic interactive proof system
trusted hardware based methods. For example, a tamper-proof, trusted hardware that is deployed on the untrusted server. The original query will be rewritten into sub-queries. All sub-queries that cannot be authenticated are processed inside the trusted hardware, while the results of the remaining sub-queries are verified by other authentication methods, e.g., the tree-based authentication methods.

### 2.3.2 Probabilistic Authentication

The probabilistic authentication methods return a probabilistic integrity guarantee (i.e., how likely the returned results are correct given that they passed the verification). The key idea of the probabilistic authentication methods is to generate the proofs of either a subset of queries or the query results of a subset of data. Thus the probabilistic approaches can be categorized into two types:

- **Query sampling** [27]: a sample of queries are picked for proof generation. For each picked query, a proof is generated for the result by executing the query on the outsourced dataset; and
- **Fake record insertion** [28], [32], [33]: The data owner inserts a set of fake records into the dataset before outsourcing it. The assumption is that SP cannot distinguish the real and fake records easily. Then for any given query \(Q\), a faithful SP should return \(Q(F)\) as a part of query results. Thus, any query result that does not include \(Q(F)\) can be caught as incorrect. The result \(Q(F)\) thus acts as the integrity proof of query answers.

### 3 Preparatories

In this section, we introduce the concept of authenticated data structure (Section 3.1) and cryptographic background (Section 3.2).

#### 3.1 Authenticated Data Structure (ADS)

In this section, we introduce two prevalent ADS structures, namely, Merkle hash tree (MHT) and Merkle B-tree (MB-tree), that are used by a number of existing ADS-based verification methods.

##### 3.1.1 Merkle Hash Tree (MHT)

One of the widely-used ADS structures is Merkle Hash tree (MHT) [37]. A MHT \(T\) is a tree in which each leaf node \(N\) stores the digest of a tuple \(t\): \(h_N = H(t)\), where \(H()\) is a one-way, collision-resistant hash function (e.g., SHA-1 [38]). For each internal node \(N\) of \(T\), it is assigned the value \(h_N = H(h_{N_1} \| \cdots \| h_{N_N})\), where \(N_1, \ldots, N_N\) are the children of \(N\), and \(\|\) is the concatenation operator. The hash value \(h_{root}\) of the root node is used as the digest of the tree. To serve the authentication purpose, a trusted party (can be the data owner himself) generates the signature of the ADS as \(Sig = Sign(sk, h_{root})\), where \(Sign()\) is a digital signature signing function, and \(sk\) is the private key. \(Sig\) is shared with the clients for authentication. Fig. 3 illustrates an example of MHT.

##### 3.1.2 Merkle B-Tree (MB-Tree)

Merkle B-tree (MB-tree) [16] enables efficient search on one-dimensional data, as it combines MHT with \(B^+\)-tree. In MB-tree, the tuples are arranged in the same fashion as standard \(B^+\)-tree. The major difference is the incorporation of hash values in the tree nodes. In particular, in the leaf node, each entry is associated with a hash value \(h = H(t)\), where \(t\) is the tuple in the entry. In the internal node, every pointer to a child node is combined with \(h = H(h_1) \cdots H(h_k)\), where \(h_1, \ldots, h_k\) are the hash values of the child node.

#### 3.2 Cryptographic Background

In this section, we introduce the cryptographic background of the signature-based and accumulation value based authentication methods. First, we explain bilinear pairing, a building block of constructing cryptographic signatures (Section 3.2.1). Then, we introduce the elliptic curve cryptography technique that is widely used to generate individual signatures for the signature-based authentication method (Section 3.2.2). Next, we introduce three types of aggregate signature methods that aggregate individual signatures into a single one (Section 3.2.3). These aggregate signature methods are popularly used for the accumulation value based authentication methods. In Section 3.2.4 we introduce the basic verification protocols for set operations. These protocols are used for authentication of multi-dimensional range queries. Finally, in Section 3.2.5, we explain the interactive proof protocol that is used for the interactive proof based authentication methods.
3.2.1 Bilinear Pairing

Bilinear pairing [39] uses a pairing between two cryptographic groups to a third group. In particular, let $G_1$, $G_2$, and $G_T$ be three groups of order $p$ (with $p$ a $\lambda$-bit prime). Also let $g_1$ ($g_2$, resp.) be the generator of the group $G_1$ ($G_2$, resp.), i.e., every element in $G_1$ ($G_2$, resp.) can be obtained by repeatedly applying the group operation on $g_1$ ($g_2$, resp.). A bilinear pairing on $(G_1, G_2, G_T)$ is a function $e : G_1 \times G_2 \rightarrow G_T$ that satisfies the following conditions:

- **Bilinearity:** $\forall a, b \in \mathbb{Z}_p, P \in G_1, Q \in G_2$, $e(P^n, Q^m) = e(P, Q)^{nm}$, i.e., the function $e$ associates pairs of elements from $G_1$ and $G_2$ with elements in $G_T$.
- **Non-degeneracy:** $e(g_1, g_2) \neq 1$;
- **Computability:** There exists an efficient algorithm to compute $e$.

Here bilinearity means linearity in both arguments of $e$. If the pairing is defined such that $G_1 = G_2$ then it is said to be symmetric, and asymmetric otherwise. More details of how the bilinear pairing is used to generate the accumulation values will be explained in Section 3.2.3.

3.2.2 Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) is a public-key cryptographic method based on the algebraic structure of elliptic curve $y^2 = x^3 + ax + b$ over finite fields, where $4a^3 + 27b^2 \neq 0$. Each combination of $a$ and $b$ produces a different elliptic curve. All points $(x, y)$ that satisfy the above equation lie on the elliptic curve. Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other tasks. An ECC signature that is 160 bits long provides comparable security to a 1024-bit RSA signature [40]. Using shorter signatures enables ECC to achieve lower storage overhead. ECC signatures have been widely used in signature-based authentication methods [18], [35].

3.2.3 Aggregate Signature

Given $n$ signatures on $n$ distinct messages, an aggregate signature scheme aggregates all these signatures into a single short signature [41]. Below we briefly explain three types of methods that generate aggregate signatures.

**BGLS Scheme.** The BGLS aggregate signature scheme (named after its four authors) [41] relies on bilinear pairing for signature aggregation. In particular, given a bilinear map $e : G \times G \rightarrow G_T$ over two groups of prime order $p$, let $s \in \mathbb{Z}_p$ be a secret key, the public key $v$ is generated as $v = g^s \in G$, where $g$ is the generator of group $G$. For any message $m_i$, let $H : \{0, 1\}^* \rightarrow G$ be a full-domain hash function. The signature $\sigma_i$ of $m_i$ is generated as $\sigma_i = H(m_i)^s$. Given $n$ signatures $\{\sigma_1, \ldots, \sigma_n\}$ where $\sigma_i = H(m_i)^s$, their aggregate signature is generated as $\sigma_{1:n} = \Pi_{i=1}^n \sigma_i$. Again, $\sigma_{1:n}$ is a member in group $G$. Upon receiving an aggregate signature $\sigma_{1:n}$, the set of messages $\{m_i\}$, and the public keys $\{v_i\}$, whether $\sigma_{1:n}$ corresponds to the messages $\{m_i\}$ can be verified by first computing $h_i = H(m_i)$ for $1 \leq i \leq n$. The messages are accepted if $e(\sigma_{1:n}, g) = \Pi_{i=1}^n e(h_i, v_i)$. BGLS has been used in [18] to aggregate the ECC signature.

**Homomorphic Linear Authentication (HLA).** HLA scheme [42] allows to check the authenticity of the messages against the generated aggregate signature without the knowledge of individual messages. In the HLA scheme, given a vector of random coefficients $\bar{c} = \{c_1, \ldots, c_n\}$, a single aggregate signature $\sigma_{1:n}$ that corresponds to $\sum_{i=1}^n c_i m_i$ is computed. The authenticity of the messages $\{m_1, \ldots, m_n\}$ can be computed from $\sigma_{1:n}$. HLA scheme has been used in [31] for generation of aggregate signatures.

**Accumulation Value.** An accumulator scheme [43], [44] allows aggregation of a large set of inputs into one constant-size value. In particular, given a set of elements $S = \{x_1, \ldots, x_n\}$, the accumulation value of $S$ is constructed as

$$\text{acc}(S) = g \prod_{i=1}^n (x_i + s),$$

where $g$ is the generator of a bilinear group, and $s$ is a randomly chosen secret value. For any pair of subsets $S_1$ and $S_2$ of $S$ such that: (1) $S_1 \cup S_2 = S$; and (2) $|S_1| + |S_2| = |S|$, it must be true that $e(\text{acc}(S_1), \text{acc}(S_2)) = e(\text{acc}(S), g)$, where $e$ is the bilinear pairing function. Based on the bilinear extension of the $\mathbb{G}$-strong Diffie-Hellman [45], the probability that any polynomial-time adversary without the knowledge of the secret key $s$ to find a different set $S_1 \neq S_2$ such that $e(\text{acc}(S_1), \text{acc}(S_2)) = e(\text{acc}(S), g)$ is negligible.

3.2.4 Verification Protocols for Set Operations

In general, the evaluation of multi-dimensional range queries can be performed by set operations, where the filtering results at each dimension are considered as a set. Next, we introduce the verification protocols for set operations. These protocols were used as the building blocks of authentication of multi-dimensional range queries that based on set operations [13], [25].

**Set Intersection.** Given a collection of sets $S = \{S_1, \ldots, S_t\}$, $I = S_1 \cap S_2 \cap \cdots \cap S_t$ is the correct intersection of $S$ if and only if: (1) $I \subseteq S_1 \wedge \cdots \wedge S_t$ (subset condition); and (2) $(S_1 - I) \cap \cdots \cap (S_t - I) = \emptyset$ (completeness condition). Papamakou et al. proposed a set intersection verification protocol to verify that $I$ is a correct intersection of $S$ [46]. For each set $S_i$, SP demonstrates $I \subseteq S_i$ by providing $\text{acc}(W_i)$, where $W_i = S_i - I$, and $\text{acc}(\cdot)$ is the function that calculates the accumulation value (Equation (1)). If $e(\text{acc}(I), \text{acc}(W_i)) = e(\text{acc}(S_i), g)$ for every set $S_i$, where $e$ is the bilinear pairing, the client is assured that the subset condition is met. The intersection completeness is proved by finding a set of polynomials $\{q_1, \ldots, q_t\}$ such that $\sum_{i=1}^t P(W_i) q_i = 1$, where $P(W_i) = \prod_{x \in W_i} (x + s)$, and $s$ is a randomly chosen value that constitutes the trapdoor in the scheme.

**Set Difference.** Papadopoulos et al. [13] designed a method for efficient authentication of set difference results based on the accumulation values. In particular, given two sets $X_1$ and $X_2$ such that $X_2 \subseteq X_1$, to demonstrate $X = X_1 \setminus X_2$, the prover constructs $\text{acc}(X)$ (Equation (1)). The verifier simply checks $e(\text{acc}(X), \text{acc}(\cdot)) = e(\text{acc}(X), g)$ for verification, where $e$ is the bilinear pairing, and $\text{acc}(\cdot)$ is the function that calculates the accumulation value (Equation (1)).

**Summation over Sets.** Given a set $S = \{x_1, \ldots, x_n\}$, Zhang et al. [25] proposed the sum verification protocol based on accumulation values to check if $\text{sum} = \sum_{i=1}^n x_i$. Let $M(S) = \prod_{i=1}^n (x_i + s) = a_0 s^n + \cdots + a_1 s + a_0$. Obviously, $\sum_{i=1}^n x_i = a_1 / a_0$. Based on this reasoning, the prover provides $a_0, a_1,
as well as proof that they are the coefficients of the smallest exponent in $M(S)$, for the purpose of verification.

### 3.2.5 Interactive Proof Protocol

An interactive proof protocol [47] allows a prover with infinite computational power to demonstrate the correctness of a statement to a polynomial time verifier. In the protocol, the verifier actively and repeatedly asks questions and receives answers from the prover. A number of works [36], [48] have investigated how to reduce the verification and communication overhead of interactive protocols to achieve practical verifiable computation.

## 4 Deterministic Approaches

In this section, we overview the five types of deterministic authentication approaches, namely tree-based solutions (Section 4.1), signature-based solutions (Section 4.2), accumulation value based solutions (Section 4.3), interactive proof based solutions (Section 4.4), and trusted hardware based solutions (Section 4.5). All these methods guarantee that the probability that the incorrect query results can escape verification of soundness and completeness is negligible. We will also discuss how to deal with data updates (Section 4.6) and how to authenticate freshness (Section 4.7).

In general, an authentication method consists of three steps: (1) verification setup by the client; (2) verification preparation by SP; and (3) verification by the client. The tree-based, signature-based and accumulation-value based authentication methods require one-time verification setup, while the interactive proof and trusted hardware based methods do not require any setup effort. However, the interactive proof method may incur expensive communication overhead between SP and the client, while the trusted hardware based methods require the hardware support. Furthermore, the proof size of the tree-based and the accumulation value based methods grows with the size of query results, while the signature-based approaches can keep proof size constant. However, the proof construction (i.e., verification preparation) by the signature-based approaches can be much more costly than the tree-based and accumulation value based methods. Finally, although the accumulation value based methods require cheaper setup cost than the tree-based methods, its verification cost can be much more expensive than the tree-based methods.

### 4.1 Tree-Based Authentication

Quite a few existing authentication approaches use the authenticated data structure (ADS) to construct the verification objects. Most of these works use different variants of Merkle hash tree (MHT) for specific query types. In this section, we discuss these authentication approaches based on the types of queries they support, namely range queries (Section 4.1.1), join (Section 4.1.2), and aggregate queries (Section 4.1.3). Besides these basic approaches, we also discuss how to optimize the authentication methods (Section 4.1.4).

#### 4.1.1 Range Queries

Devanbu et al. [21] initialized the research on authentication of SQL query evaluation. They use Merkle hash tree (MHT) to construct the ADS from the database in which the records are sorted on a set of attributes. During query execution, SP traverses the MHT and gathers the respective nodes in the MHT to construct the VO. For instance, given a single-dimensional range query $Q$, the VO includes the lowest common ancestor in MHT that cover the records in $Q$ and their nearest lower/upper bound nodes, as well as the nodes in the proximity subtrees that contain the boundary nodes occurring consecutively to the smallest (largest) element of $Q$. Consider the MHT in Fig. 4 as an example. Assume the leaf nodes (colored gray) $N_6$ and $N_7$ fall in the query range. The nearest lower (upper, resp.) bound node is $N_5$ ($N_8$, resp.). For VO construction, SP identifies: (1) node $N_{58}$, which is the lowest common ancestor of nodes $N_6$ and $N_7$; and (2) the proximity tree that includes $N_{58}$ and the two paths from $N_{58}$ to $N_5$ and to $N_8$. SP includes the hash value of $N_{58}$ and the proximity trees in VO. Soundness of $Q$ is verified by checking if $N_6$ and $N_7$ are within the query range. The completeness of $Q$ is verified by re-constructing the hash of $N_{58}$. If the result is complete, the client should be able to re-construct the hash of $N_{58}$, since there is no record at the left side of $N_5$ or the right side of $N_6$ that falls in the query range.

One weakness of the MHT-based method [21] is that the VO size depends on the size of the query answers and the outsourced dataset, which may incur expensive overhead for large query answers and/or dataset. To eliminate this dependence on data and size of query results, Pang et al. [20] designed a new ADS named verifiable B-tree (VB-tree). VB-tree is similar to MB-tree (Section 3.1.2) that it is a MHT built on top of the $B^+$-tree by adding a digest on every $B$-tree node computed by a cumulative and commutative hash function. During VO construction, SP identifies the smallest subtree (called enveloping tree) in VB-tree that covers all the records in the query results. For a given range query, the VO of its results includes: (1) the signed digest for the node at the root of the smallest subtree (called enveloping tree) in VB-tree that covers all the records of the query; and (2) the signed digest for each node in the enveloping subtree that represents those branches that do not overlap the result. The verification of soundness, and completeness is similar to [21]; the client re-constructs the signed digest of the root of the VB-tree, and matches it with the local copy that is shared by the data owner. Unlike the MHT-based solution [21] whose VO contains the digests of all the nodes all the way to the root of the tree index, the VO by [20] only needs to contain proofs for the smallest subtree that envelops the query result. Therefore, the VO size grows...
linearly with the size of the query results, but is independent of the data size.

Although VB-tree [20] eliminates the complexity dependency on data size, it may suffer from the strong assumption that SP must be trusted to some degree [17]. Furthermore, VB-tree uses a one-way function based on modular exponentiation, which was shown infeasible to handle large answer sizes [17]. Therefore, Nuckolls [17] proposed a new ADS named Hybrid Authentication Tree (HAT), which incorporates fast hash functions with an efficient RSA based one-way quasi-commutative accumulator [43]. The key idea is to use Merkle hashing to certain nodes of the tree and then use the accumulator to verify the values of those nodes, thus eliminating the hashing along remaining path to the root. The set of nodes to consider for VO construction is the set of canonical covering roots (CCR), which is the set of nodes with disjoint sub-trees whose leaves are the exact answer to the range query. The node pair with the smallest sub-tree is the covering pair for the range, or covering node in case there is only one. Fig. 5 shows an example of the CCRs and covering nodes in a HAT. Soundness of query results are verified via the accumulation value computed from the Merkle hashing of the covering nodes. Completeness verification is performed in two steps: (1) it verifies if the boundaries of the returned nodes are complete, and (2) it verifies that the leaves of the two covering nodes form a continuous range in the tree. These two steps are verified by re-constructing the hash values of the covering nodes, which were included in VO.

4.1.2 Join Queries

Yang et al. [34] designed three authenticated join algorithms: (1) Authenticated Indexed Sort Merge Join (AISM), which utilizes a single ADS in one of the base relations, (2) Authenticated Index Merge Join (AIM) that requires an ADS for both relations, and (3) Authenticated Sort Merge Join (ASM), which does not need an ADS. Consider the join of two tables \( R \) and \( S \) on the attribute \( J \). The AISM algorithm relies on the MB-tree and the ranked-list for authentication of join results of \( R \) and \( S \). In the pre-processing step, the outer table \( R \) is sorted on the join attribute \( J \) with the output as a rank list. The rank list outputs the verifiable order by which the client follows to verify the signature of records in the results. The query of finding the join records of \( S \) in \( R \) is equivalent to a single range query \( Q = \sigma_{c_l, c_u}^J R_l \) where \( c_l \) and \( c_u \) are the lower- and upper-bound value of \( S \) on the join attribute \( J \). The join result is authenticated by utilizing the MB-tree [16] of \( S \) as the ADS and the authentication method is similar to [21]. Fig. 6 shows an example of the AISM algorithm. Consider the given query \( R \Join_{s=s_j} S \) whose join result is \( \{(\Omega_R[1], s_1), (\Omega_R[2], s_{11}), (\Omega_R[3], s_{11})\} \), where \( \Omega_R \) is the rank list of tuples in \( R \), and \( \Omega_R[i] \) denotes the \( i \)th record in the rank list. Note that \( \Omega_R[2] \) and \( \Omega_R[3] \) share the same value on the join attribute \( J \). Hence both of them are matched with \( s_{11} \). Based on the MB-tree \( T_S \) of \( S \), for each record \( \Omega_R[i] \) in \( R \) that has a matching record \( s_j \) in \( S \), AISM includes its left and right boundary records in \( S \), as well as the matching record \( s_j \) and other necessary nodes to reconstruct the root of \( T_S \). While for those records in \( R \) that do not match any record \( s \), e.g., \( \Omega_R[1] \), \( \Omega_R[5] \) and \( \Omega_R[6] \) in Fig. 6, AISM only needs to incorporate those nodes in the search path. A nice property of AISM is that it only traverses \( T_S \) once from left to right when constructing the VO.

The AIM algorithm aims to save the communication cost by constructs two MB-trees for \( R \) and \( S \) respectively. Based on the two MB-trees \( T_R \) and \( T_S \), SP alternatively searches the matching records on each ADS and prepares the VO. Unlike AISM and AIM, ASM does not require any ADS at all. It sorts the data values of the join attributes \( J \) of \( R \) and \( S \) as two rank list respectively. The two sorted tables \( R' \) and \( S' \) are merged as a single rank list, in which the records that can be joined are marked. Then it creates a bitmap \( B_R \), in which the bit of the records of \( R \) that have join partners in \( S \) is set as 1, otherwise 0. Both signatures of \( R \) and \( |R| \) and \( B_R \) are added to VO. The same process repeats for \( S \). The client verifies soundness and completeness of the join result by checking the bitmap values of the received results against the VO.

4.1.3 Aggregate Queries

Li et al. [23] showed that the authentication of SUM queries is equivalent to the authentication of prefix sums [49]. To authenticate prefix sum, Li et al. designed a new ADS structure named the authenticated prefix sums tree (APS-tree), which takes the format of a multi-way MHT. Each leaf entry is converted into a base-f number. To construct the VO of a given query \( Q \), SP returns the \( 2^d \) corner prefix sum values as the part of VO, where \( d \) is the number of dimensions of the dataset. Besides, the VO contains the hash values of the APS-tree nodes that are needed to authenticate the sum results, as well as the encoding of the path for each node. Based on the VO, the client authenticates the soundness of the SUM result by verifying the prefix sum values. In particular, the client authenticates each of the \( 2^d \) elements of the answer set by computing the hash of the root for each path. Then it compares the hash with the local signature generated by the data owner. Once the \( 2^d \) corner prefix sum values are verified, the soundness of the aggregation result is verified by aggregating the prefix sum values and
comparing the result with the sum value returned by the server. Li et al. [23] also designed another ADS named authenticated aggregation B-tree (AAB-tree) for the dynamic case. The AAB-tree is an MB-tree with each node associated with an aggregate value as the sum of the aggregate values of its children, and a hash value over the concatenation of both the hash values and the aggregate values of the children. An example of AAB-tree is shown in Fig. 7. The AAB-tree can be used for authentication of one-dimensional aggregate queries in a dynamic setting since the owner can easily issue deletions, insertions and updates to the tree as for the normal B+-tree. It can be easily extended to deal with the multi-dimension aggregate queries in the similar way as the APS-tree. Other than COUNT and AVG, AAB-tree supports authentication of MIN and MAX as well, by replacing the SUM aggregate in each entry with the MIN/ MAX aggregate.

4.1.4 Optimization of Verification

In this section, we discuss two types of optimization methods that can reduce the overhead.

Adding a Trusted Entity. Most of the existing ADS-based verification methods involve the data owner into the verification setup for ADS construction, which is not feasible for the data owner who has limited computational resources. Furthermore, the VO is typically large, which may bring significant communication overhead. To address these two drawbacks, Stavros Papadopoulos et al. [15] separate authentication from query execution by exploiting a trusted entity (TE). The high-level idea is that the data owner sends her dataset to TE. TE generates the digest of each record by using a one-way, collusion-resistant hash function. When the client receives the result of a range query from SP, the client sends the query to TE. TE evaluates the query on the dataset received from the data owner, and produces a verification token, which is constructed by applying the exclusive-OR (XOR) operator on the digests of records in the query result. TE transmits the verification token to the client. The client computes the XOR of the digests of the records in the returned results by SP, and matches it against the verification token by TE. To facilitate the authentication process, [15] includes the design of a new ADS named XOR B-Tree (XB-Tree) that integrates the B-tree with XOR values. The XB-Tree is very similar to the B-tree; the difference is that each entry of the XB-Tree is associated with a bit string that represents the results of XOR operator. An example of XB-Tree is shown in Fig. 8. Search on the XB-Tree and VO construction are very similar to those on the MB-tree.

Separating ADS from Data Index. So far most of the techniques incorporate ADS (e.g., MHT) with a data index; SP has to traverse the ADS and gather the respective nodes to form the VO. Mouratidis et al. [22] designed a novel scheme called Partially Materialized Digest scheme (PMD) that separates MHT from the data index. Instead, it uses a main index (MI) for querying, and a separate digest index (DI) for query authentication. The MI is a standard B+-tree on key. Range queries are evaluated on MI in the same way as on the B+-tree. The DI is constructed as follows. First, for each leaf MI node $E_i$, a lower MHT is built on top of it. Then the upper MHT is built over the roots of the lower trees. The root of the upper tree is signed with the owner’s private key. Similar to [21], for a given range query $Q$, besides the records that satisfy the range of $Q$, SP identifies two boundary records, $p$ and $p^+$, falling immediately to the left and to the right of the query range. The VO is constructed by using the DI only. It contains (1) the signed DI root, (2) all left sibling hashes to the path of $p$, and (3) all right sibling hashes to the path of $p^+$. To verify if the received result is correct and complete, the client combines $R$ with the components (2) and (3) of VO to reconstruct the root of DI. If the signature of the root of DI (i.e., component (1) of the VO) matches the locally computed root hash, the result is correct and complete. By separating these two indices, PMD significantly outperforms the performance of MB-tree [16] in terms of query response time, storage overhead and index construction cost.

4.2 Signature-Based Authentication

An alternative verification approach is to generate the VO in the format of a cryptographic signature. A new integrity authentication method that relies on signature aggregation for VO construction is designed for verification of soundness and completeness of query results. The key idea of signature aggregation is to generate an authenticated chain from the signatures of records, and outputs the aggregate signature of the authenticated chain as the proof. In this section, we overview the authentication methods based on signature aggregation by considering the types of queries they support, namely range queries (Section 4.2.1), join (Section 4.2.2), and aggregate queries (Section 4.2.3).

4.2.1 Range Queries

Mykletun et al. [18] (appeared in 2004, published in 2006) proposed the first signature-based method for range query authentication. By their approach, the data owner constructs a standard RSA signature [50] for each record, and sends these signatures together with the dataset to SP. SP uses condensed RSA [50] to compress the RSA signatures of all the hit records in the query results into a single signature. Alternatively, the owner constructs an ECC signature for every record, and SP uses the BGLS scheme (Section 3.2.3)
to aggregate them. Then the aggregate signature is sent to
the client together with the query results as its VO. The cli-
ent verifies the soundness of the returned query results by
verifying the returned records against the single signature.
The major advantage of this method is that the VO size is
significantly reduced by combining multiple record signa-
tures into one single signature.

Although effective, the aggregate signature [18] cannot ver-
ify the completeness of the query results. Narasimha et al. [30]
extended [18] by proposing Digital Signature Aggregation and
Chaining (DSAC), which combines signature aggregation with
signature chaining for the authentication of range queries.
Unlike [18] that aggregates the signatures of individual
records, DSAC [30] constructs the VO from the signature of
each individual record along with its immediate predecessor,
i.e., \(\text{sign}(H(t_i||t_{i-1}))\), where \(t_{i-1}\) is the immediate predecessor of
\(t_i\) on the search attribute. By including the predecessor in
the signature, it forms a chain of records ordered on the search
attribute. Fig. 9 shows an example of range query processing
in [30]. The tuples between \(t_a\) and \(t_b\) fall into the query range
(corresponding to the gray nodes in Fig. 9). SP returns the
result set, as well as the two boundary tuples, i.e., \(t_{a-1}\) and \(t_{b+1}\).
In addition, SP constructs an aggregate signature constructed
from the signatures of \(t_a\) to \(t_{b+1}\). The soundness and complete-
ness are verified by checking if the set of records received in
the result indeed form a valid chain, if \(t_a\) and \(t_b\) is within the
query range, and if \(t_{a-1}\) and \(t_{b+1}\) are out of the query range.

The aggregate signature [18] and DSAC [30] only support
single-dimension range queries on the key attribute. Cheng
et al. [24] designed a new authentication method for multi-
dimensional range query evaluation. They combined aggre-
gate signature with KD-tree [51] to construct the ADS named
VKD-tree. Before outsourcing, the data owner constructs the
KD-tree of the dataset to get data partitions. Note that each
data partition involves multiple dimensions of the dataset.
For each partition \(P\) that is included in the query range, SP
returns an aggregate signature of all the enclosed records and
the partition. The client verifies the completeness by using the
aggregate signature. For each partition \(P\) that is out of the
query range, SP returns the bounding records and the signa-
ture of \(P\). The signature of \(P\) is constructed as \(\text{sign}(t_0||t_{k+1}|k)\),
where \(t_0\) and \(t_{k+1}\) are the bounding records, and \(k\) is the
size of the partition. The client verifies the completeness of
the query result by checking the soundness of the bounding
records against the signature of \(P\). If the bounding records are
correct, then the client is convinced that no record in \(P\) is
included in the query result. For those partitions that overlap
with the query range, they are split into sub-partitions that
non-overlap/overlap with query ranges. These sub-partitions
are verified by using the aforementioned two approaches.

4.2.2 Join Queries
Pang et al. [29] devised a new signature-based scheme for
range and join queries. It constructs the VO from the signature
of each individual record along with both of immediate pre-
decessor and successor, i.e., \(\text{sign}(H(g(t_{i-1})||g(t_i)||g(t_{i+1})))\),
where \(t_{i-1}\) and \(t_{i+1}\) are the immediate predecessor and suc-
cessor of \(t_i\) on the search attribute respectively, \(\text{sign}\) is the digital
signature signing function (e.g., RSA or DSA [50]), \(H\) is the
hash function, \(||\) is the concatenation operator, and \(g(t_i)\) is the
digest of \(t_i\). [29] used an iterative hash function to ensure the
authentication method is secure against cheating by the SP.
The proposed scheme can support joins on primary key and
foreign key, by using signatures on the key attribute of a
relation to generate proof of the completeness of query results
from that relation. Fig. 10 shows an example of the authentica-
tion method by [29]. Consider a given query \(R \bowtie_{B,B=S,A} S\),
where \(A\) is the primary key attribute in \(S\), and \(B\) is the corre-
sponding foreign key attribute in \(R\). Thus every \(r_i \in R\) has at
least one matching tuple in \(S\). In the example shown in
Fig. 10, \(r_i\) is matched with \(s_2\) and \(s_3\). Thus in the VO, SP
includes the signatures of \(s_2\) and \(s_3\), as well as the two bound-
ary tuples \(s_1\) and \(s_4\). The client verifies the join result on \(r_i\) by
using the signatures.

4.2.3 Aggregate Queries
Zheng et al. [31] designed the signature-based authenti-
cation approach named AuthODS for aggregate queries. It
utilizes the Homomorphic Linear Authentication (HLA)
(Section 3.2.3) to construct the aggregate signature for
authentication of SUM queries. To prove \(w = t_a + t_{a+1} + \cdots + t_b\),
SP prepares the proof \(\sigma_{ab}\) by executing the
\(\text{HLA}_{\text{agg}}(\vec{c}, T_{\text{agg}})\) protocol (Section 3.2.3), where \(\vec{c}\) is a vector
of 1s, \(T_{\text{agg}} = \{\sigma_1, \ldots, \sigma_b\}\), and \(\sigma_i\) is the signature for record
\(t_i\). The signature \(\sigma_{ab}\) is for the aggregated message with
regard to the coefficients in \(\vec{c}\), i.e., \(\sum_{i=a}^{b} c_i\). Therefore, the
client can verify soundness and completeness of \(w\) by executing the
\(\text{Vrfy}(pk, w, \sigma_{ab})\) protocol (Section 3.2).

4.3 Accumulation Value Based Verification
Most of the ADS-based authentication approaches share one
weakness: they have to construct one ADS for each possible
combination of dimensions for authentication of multi-
dimensional range queries. This makes the number of ADS
scale exponentially with the number of dimensions. The
accumulation value based verification methods address this
weakness by avoiding the construction of a large number of
ADS structures for query authentication. In this section, we
discuss the accumulation value based authentication meth-
ods for range queries (Section 4.3.1), join and aggregate
queries (Section 4.3.2).

4.3.1 Range Queries
Papadopoulos et al. [13] designed an authenticated multi-
dimensional range query protocol (AMR) by integrating the
accumulation values with MHT. They use the authentication
protocol of set intersection and difference (Section 3.2.4) as the building blocks. The authentication method consists of two steps. Let \( R \) be the query result. The first step is that SP computes the set of hash values \( R_i \) of the records that satisfy the range selection constraint on dimension \( i \). It also computes the proof \( \pi_{R_i} \), which is used to verify the soundness of \( R_i \) (i.e., if \( R_i \) is the set of records that match the range constraint on dimension \( i \)). The second step is that SP computes \( R = \bigcap R_i \) and generates the proof \( \pi \) of \( R \) by the set intersection verification protocol on \( \pi \) (Section 3.2.4). The client uses the same verification protocol to verify the soundness and completeness of \( R \). An example of the verification process of AMR [13] is displayed in Fig. 11. The range query issues the selection condition on the attributes \( A_i \) and \( A_j \), \( R_i \) and \( R_j \) are the set of tuples that satisfy the condition on \( A_i \) and \( A_j \), respectively. The client first verifies the correctness of \( R_i \) and \( R_j \), and then inspects \( R \) by using the set intersection protocol on \( R_i \) and \( R_j \).

### 4.3.2 Join and Aggregate Queries

Zhang et al. [25] designed a novel authentication system named IntegriDB to accommodate a wide range of SQL queries, including multi-dimensional range queries, JOIN, SUM, MAX/MIN, COUNT, and AVG. The authors designed a sum verification protocol (Section 3.2.4) and a new ADS named Authenticated Interval Tree (AIT-tree), and facilitates them as the key components of IntegriDB. The AIT-tree is designed for key-value pairs. Given a set of key-value pairs \( S = \{(k, v)\} \), the AIT-tree is a binary-structured tree constructed from \( S \). For each key-value pair \((k, v) \in S\), it corresponds to a leaf node in the AIT-tree, which stores a triplet \((k, v, h)\) with the hash value \( h = H(k||v) \). The leaf nodes are arranged in the order of the keys. The value stored at an internal node \( N \) corresponds to the accumulation value (Section 3.2.4) of the leaves in the subtree rooted at \( N \). The verification of one-dimensional range query result is similar to [21]. The verification of multi-dimensional range queries and join queries is performed by the authentication protocol of set intersection (Section 3.2.4), by considering the results of each individual dimension/table as a set. Regarding the aggregation queries, they are authenticated by different methods. SUM results are verified by using the sum summation protocol (Section 3.2.4). In order to facilitate the verification of COUNT operations, the data owner creates an additional attribute \( A'_j \) for each attribute \( A_j \). The value of \( A'_j \) of the record \( t_i \) is set as \( t_i[A'_j] = t_i[A_j] + 1 \). Obviously, \( \text{COUNT}(A_j) = \text{SUM}(A_j) - \text{SUM}(A_j') \). Thus the soundness of \( \text{COUNT}(A_j) \) can be verified by the proof of \( \text{SUM}(A_j) \) and \( \text{SUM}(A_j') \). MAX/MIN queries are transformed to single-dimensional range queries.

### 4.4 Interactive Proof Based Verification

Zhang et al. [26] designed vSQL, a system for verifiable SQL queries over dynamic outsourced databases. vSQL combines two different approaches, namely information-theoretic interactive proof system [36] and a novel scheme for verifiable polynomial delegation that can provide auxiliary inputs for the proof. vSQL first translates SQL queries into arithmetic circuits. Then it relies on an information-theoretic interactive proof system named the CMT protocol [36], which allows a client to verify that \( y = C(x) \), where \( x \) is the outsourced data, and \( C \) is a circuit corresponding to the clients query. It is not feasible to apply the CMT protocol directly on SQL query evaluation, since the client may not have access to \( x \) (i.e., the outsourced database). To address this problem, vSQL designs a new polynomial-delegation protocol, which allows the client to be able to evaluate a certain multivariate polynomial \( p_x \) that depends on \( x \) (but not on \( C \)) at a random point. In particular, the data owner treats the database \( D \) as an array of elements, and computes the multi-linear extension \( D' \) of \( D \). The data owner uses the polynomial-delegation protocol to generate the commitment \( \text{com of } D' \), and sends \( \text{com} \) to the client and sends \( D \) to SP. At the query evaluation phase, the client verifies the query result following the CMT protocol. The verification of all layers except the last one follows the CMT protocol between the client and SP. During the last step of the CMT protocol, vSQL evaluates at a random point of \( D' \) that is picked by SP. Since the client does not have access to \( D' \), it uses the commitment \( \text{com} \) via the polynomial-delegation protocol. The protocol assures that no computationally bounded adversarial server can convince the client to accept an incorrect result with non-negligible probability.

### 4.5 Trusted Hardware Based Verification

Most of the existing work discussed query verification through software mechanisms. An alternative approach is to rely on the tamper-proof, trusted hardware deployed on the untrusted server. Bajaj et al. [14] design a system named CorrectDB that utilizes the trusted hardware such as the IBM 4764 [52] coprocessor (SCPU) to provide secure execution in an untrusted environment. The root hash of the MHT-based ADS is stored on SCPU. This can eliminate the generation of digital signatures of root node hash. The original query is rewritten into a set of sub-queries. The query rewriting process ensures that the processing within the SCPU is minimized, and any intermediate results generated by SP can be validated by the SCPU using the ADS. Any operation executed by SP that cannot be authenticated are processed inside the SCPU. CorrectDB [14] supports range queries, projections, joins, aggregations, grouping, and ordering. When there is any data update, the MHT-based ADS is updated accordingly by the SCPU.

### 4.6 Dealing with Data Updates

In this section, we discuss how the data updates are dealt with by the authentication methods.
In general, for the tree-based authentication methods, the ADS has to be updated according to the data updates. The leaf nodes and their hash values in the ADS are the first to be updated. The root signature is updated by propagating the update on the leaf nodes along their paths up to the root. The updates of both data and ADS are sent back to SP. The digests of ADS root signature have to be re-distributed to all clients. Li et al. [16] are one of the first works that designs dynamic ADS for query authentication. They design the EMB-tree, which consists of regular $B^*$-tree entries augmented with an embedded MB-tree (Section 3.1.2). Each EMB-tree node consists of a triplet $(k_i, p_i, h_i)$, where $k_i$ is the key of the node, $p_i$ is the pointer to the embedded Merkle tree at this node, and $h_i$ is the hash value of the embedded tree. An example of EMB-tree is shown in Fig. 12. When there are updates on the data, only the path from the affected leaf (i.e., the updated data record) to the root is updated. The VO construction and query authentication procedure based on the EMB-tree is similar to the MHT-based method [21].

The naive approach for handling batch updates would be to do all updates to the ADS one by one and update the path from the leaves to the root once per update. Obviously this naive method may lead to expensive cost of performing unnecessary hash function computations on the predecessor path when a large number of updates affect a similar set of nodes (e.g., the same leaf). To optimize the update cost on the ADS, Li et al. [16] suggested to recompute the hashes of all affected nodes only once, after all the updates have been performed on the tree.

**Signature-Based Methods.** Any update to a record requires to re-compute the signed digests of the record as well as its neighboring ones. Narasimha et al. [30] provide a detailed protocol that supports efficient updates on the digest. Briefly speaking, upon receiving the data owner’s request to insert a new record $t_r$, the server fetches its neighbors $t_{l_r}$ and $t_{r+1}$, which are $t_r$’s immediate predecessor and successor respectively, and sends them back. To prove the correctness of the neighbor records, SP constructs an aggregate signature $σ$ of them. After verifying $t_{l_r}$ and $t_{r+1}$ against $σ$, the data owner constructs the signature of $t_r$ with $\text{sign}(H(t_r||t_{l_r}))$, updates the signature of $t_{r+1}$ with $\text{sign}(H((t_{r+1})||t_r))$, and returns them to the server.

**Accumulation Value Based Methods.** When inserting/deleting a record, the data owner needs to update the accumulation values of these records [25]. Papadopoulos et al. [13] designed an update-efficient accumulation value based method. In particular, the data owner creates $b$ non-overlapping buckets, and constructs two levels of accumulation values, i.e., the bucket-level accumulation values and the record-level accumulation values inside a bucket. A new record $t_r$ triggers the update of the accumulation values for all records which are inside the same bucket and before $t_r$, and for all buckets before the bucket that $t_r$ belongs to.

**Interactive Proof Based Methods.** The interactive proof based method [26] supports efficient data updates by separating the computation of the update from its verification. SP has to update the digest $\text{com}$ according to the data updates. The client can provably evaluate if $\text{com}$ is updated by using the polynomial-delegation scheme.

**Trusted Hardware.** CorrectDB [14] allows the data owner to issue an update query directly to the SCPU. All updates are then performed by the SCPU, and thus eliminating the update overhead at the side of the data owner.

### 4.7 Freshness Authentication

Most of the works discussed so far only consider the verification of soundness and completeness of the query results. In this section, we review the existing studies on verification of freshness of the query results, i.e., these results are obtained by executing the queries over the most up-to-date data. A straightforward solution is to extend the proof-based solutions to provide freshness verification. The key challenge of such an extension is to ensure that the signatures attached to individual records are indeed constructed from the most up-to-date data. Then the client can verify the freshness of the query results from the constructed proof of the query results in the similar way as the existing proof-based methods, as long as the proof is constructed from the up-to-date signatures, for example, either by aggregate signature or by the tree-based ADS. Li et al. [16] first raises the issue of query freshness. They provide a number of solutions, including publishing a list of revoked signatures, including the time interval of validity as part of the signed message and reissuing the signature after the interval expires, and using hash chains to confirm validity of signatures at frequent intervals. They point out that any of these approaches can be applied directly on the tree-based solutions to update the single signature of the root of the tree. Each data update will require re-issuing one signature only. Xie et al. [35] design a scheme that associates the signature $S_i$ (e.g., the aggregate signature or the root signature of MHT tree) with a certificate $\text{Certificate}_{c_i}(S_i, t)$ in the outsourced database, to indicate that the current signature at time $t$ is $S_i$. When a client retrieves the query result, it also retrieves the signature and verifies if it is for timestamp $t$. Pang et al. [19] combined signature chaining with timestamps. The key idea is that the timestamps are embraced into the calculation of hash values of records. Thus the signature of a record $t_r$ is calculated as $\sigma_r = \text{sign}(H(i||t_r||t_{r-1}||t_{l_r}||t_{r+1}||t_{r+s}))$, where $\text{sign}()$ is an ECC signature signing function, $H()$ is a collision-free hash function, and $t_{r-1}$ is the immediate left neighbor of $t_r$ along an ordering attribute, and $t_s$ denotes the timestamp of the last update for record $t_r$. The reason of using ECC instead of RSA is because ECC can achieve the same security level by using much shorter signatures. For example, an ECC signature that is 160 bits long provides comparable security to a 1024-bit RSA signature.

![Fig. 12. An example of EMB-Tree node](image-url)
Based on the this signature scheme, the proof is constructed in the similar way as [30]. The proof can be used to verify soundness, completeness, and freshness.

5 Probabilistic Authentication Methods

In this section, we present the two types of probabilistic authentication methods, namely the query sampling approaches and the fake record based approaches, for the verification of soundness and completeness of range queries (Section 5.1) and join queries (Section 5.2). We also discuss how to deal with data updates (Section 5.3) and authentication of freshness (Section 5.4).

5.1 Range Queries

Sion [27] designed the first probabilistic verification approach for query execution on outsourced databases based on query sampling. Its key idea is to generate query execution proofs; for each executed batch of queries the database service provider is required to provide a strong cryptographic proof that provides assurance that the queries were actually executed correctly over the outsourced dataset. Specifically, given a batch of $b$ queries $\{Q_1, \ldots, Q_b\}$, the client first generates $r > 1$ random numbers $\{x_1, \ldots, x_r\}$, where $1 \leq x_i \leq b$. Then, according to the random indexes, the client computes the query results $\{\rho(Q_{x_1}), \ldots, \rho(Q_{x_r})\}$ for the picked queries. For each query, it generates a challenge token $C(Q_{x_i}) = (H(\epsilon), \rho(Q_{x_i}), \epsilon)$, where $\epsilon$ is randomly generated. The client sends the query batch as well as the challenge tokens (as the query execution proof) to the SP. SP returns the query results and the query execution proof $\{x'_1, \ldots, x'_r\}$. The results pass the verification only if $\{x_1, \ldots, x_r\}$ matches $\{x'_1, \ldots, x'_r\}$. Due to the one-way non-invertible cryptographic hash function, SP has to execute the queries over the target dataset to obtain the correct random indexes. The probability that SP gets all the random indexes by executing $w < b$ queries is

$$P_r(w, r) = \frac{1}{\binom{b}{w}}.$$  

The client can control the escape probability that SP passes the verification to be below a given threshold by adjusting $r$, i.e., the number of queries with challenge tokens. According to the experimental results, an assurance level of 5 percent escape probability requires the client to prepare 25 percent queries with challenge tokens.

Xie et al. [28] proposed the first probabilistic authentication framework that verifies the soundness, completeness and freshness of the query results on outsourced encrypted databases based on fake records. The key idea is that the client creates a small set of fake records $\Delta$, and inserts $\Delta$ into the original database $D$. SP processes queries over the database with $\Delta$, assuming it cannot distinguish real records from the fake ones. The correctness of a query result $R_Q$ is evaluated by checking if the returned result $R^\Delta$ contains $Q(\Delta)$, where $Q(\Delta)$ is the results of evaluating $Q$ on $\Delta$. If it does not, the client believes that the results violates the correctness requirement with 100 percent certainty. Otherwise, the client trusts the correctness of $R^\Delta$ with a probability. Formally, assume the original dataset $D$ has $N$ records, and $K$ fake records are inserted into $D$. The probability that an attacker can delete $m$ records without being caught is

$$\prod_{i=0}^{m-1} \frac{N-i}{K+(N-i)}.$$  

The empirical study in [28] shows that for a dataset of $N = 1,000,000$ records, when the fake records are more than 10 percent of the original data, and more than 50 records are deleted, it is close to impossible for the attacker to escape from being caught by the probabilistic approach.

5.2 Join Queries

Di Vimercati et al. [32], [33] consider the join of two relations, $B_l$ and $B_r$, hosted at two storage SPs $S_l$ and $S_r$. The integrity of $B_l$ and $B_r$ is verified by inserting three types of fake records into the outsourced dataset: (1) markers records inserted to both $B_l$ and $B_r$; (2) twins records, which are a portion of real records that each of the storage SPs duplicates locally before sending it to SP; (3) salts/buckets records that are inserted to destroy recognizable frequencies of combinations in one-to-many joins. Salts are used on the records at the many-side of the join so that occurrences of a same value become distinct. Meanwhile the salted replicas are created at one-side of the join to create the corresponding matching. Buckets allow multiple occurrences of the same (encrypted) value at the many-side of the join, requiring that all the values have the same number of occurrences. The client can verify the result correctness regarding the markers and twins. The probability that no marker is omitted is $(1 - \frac{1}{f})^m$, where $f$ is the cardinality of a relation with $t$ twin pairs and $m$ markers, and $o$ is the number of original records that are omitted without being detected. The probability that for each twin pair, either both records are omitted or both are preserved is $(1 - 2 \frac{1}{t})^m$.

5.3 Dealing with Data Updates

For the probabilistic verification methods based on query sampling, the data owner can choose new samples of queries. For the verification methods based on fake records, the data owner can generate new fake records according to data updates. The digest values of these new records can be readily calculated at the client side.

5.4 Freshness Authentication

There are several challenges to extend the probabilistic authentication methods to support freshness verification. First, the client must know what are the most up-to-date fake records in the outsourced database. This means that the client must be aware of all insertions and deletions of the fake records starting from the beginning. Second, it is important to ensure the fake record based scheme is provably secure, i.e., the attacker cannot distinguish fake operations from real operations. To address these challenges, Xie et al. [35] devise a mechanism to make all fake operations deterministic so that the client can derive the latest status of the fake records in the outsourced database. Specifically, the fake operations scheme is defined as $(FS,T,H)$, where $FS$ is a set of functions to generate the fake records, $T$ is a function to decide when to submit the fake operations and $H$ is a
function to decide when a function in FS is used to create fake records. The paper shows that the scheme is a provably secure scheme, i.e., its security can be reduced to the underlying encryption primitives.

6 COMPLEXITY, PERFORMANCE, AND PRACTICAL ISSUES

6.1 Evaluation Metrics and Parameters
Typically, the performance of the authentication methods is measured according to the following six metrics [16]: (1) the computation overhead for the owner, (2) the owner-SP communication cost, (3) the storage overhead for SP (e.g., ADS size), (4) the computation overhead for SP, (5) the client-SP communication cost, and (6) the computation cost of verification for the client. The input parameters for evaluation include the size of the outsourced dataset, number of attributes of the dataset, number of queries, the selectivity of queries (i.e., the percentage of records that satisfy the selection conditions of queries), the probabilistic integrity guarantee (for probabilistic authentication methods), and the structure parameter for ADS (e.g., fanout and height of MHT).

6.2 Complexity Analysis and Comparison

6.2.1 Deterministic Approaches
In this section, we discuss the complexity of the main deterministic authentication methods that are discussed in this survey. The summary of complexity analysis is present in Table 2.

In general, the tree-based, signature-based and accumulation-value based authentication methods require one-time setup. At the setup phase, the tree-based methods have to construct ADS, while the signature-based and the accumulation-value based methods construct the signatures/accumulation values of individual records.

For most of the tree-based approaches (e.g., [15], [16], [17], [21], [23]), the setup time and storage overhead grow polynomially with the size of the database and the number of attributes. The only exception is APS-tree [23], whose setup time and storage overhead grow exponentially with the number of attributes. The VO construction time of the tree-based approaches mostly grows linearly with the size of the database and the number of attributes. The setup cost of IntegrityDB [25] is polynomially with the size of the database and the number of attributes, its verification time of the tree-based approaches mostly grows linearly with the data size and query selectivity (e.g., [15], [21], [22], [23]), or grows linearly over the size of the query results (e.g., [15], [20]). The verification time of the tree-based approaches mostly grows linearly with the size of query results and polynomially with the size of the database (e.g., [15], [16], [21], [22], [23]), or grows linearly with the size of the query results (e.g., [15], [20]). The verification time of the tree-based approaches mostly grows linearly with the size of query results and polynomially with the size of the database (e.g., [15], [16], [21], [22], [23]).

For most of the signature-based approaches (e.g., [18], [19], [24], [29], [30], [31]), the setup overhead, proof construction and verification cost increases linearly to the data size. The proof size either remains constant [18], [19] or grows linearly with the data size and query selectivity (e.g., [24], [29], [30], [31]).

The cost of setup, storage, update, proof construction, verification, as well as proof size, of the accumulation value based authenticated protocols [13] grows linearly with the number of attributes. The setup cost of IntegrityDB [25] is quadratic to the number of attributes, while the complexity of verification is similar to that of [13].

The interactive proof based approach can lead to a significant performance speedup at SP side compared with the generic verification approaches [53], [54]. Its performance is also comparable to a deterministic approach [25] that only supports a restricted subclass of SQL queries.

To summarize, since the VO size of the signature-based methods can be constant, its communication and storage overhead can be smaller than the tree-based approaches by one order of magnitude when query result includes more than 1,000 tuples. On the other hand, the computational overhead at SP side of the signature-based method is more expensive than the tree-based approaches, since construction of the signature-based proof is costly (computing even a single cryptographic trapdoor requires a high number of CPU cycles, which can cost up to 30,000 picocents [14], [55]). Since SP is much more computationally powerful than the

### Table 2

| Method         | Paper                                  | Setup            | VO Construction | VO size       | VO Verification |
|----------------|----------------------------------------|------------------|-----------------|---------------|-----------------|
| Tree-based     | MHT [21] APS-Tree, AAB-Tree [23]       | $O(mn)$          | $O(q + \log n)$ | $O(q + \log n)$ | $O(q + \log n)$ |
|                | AISM,AIM,ASM [34]                      | $O(n)$           | $O(nq)$         | $O(nq)$       | $O(nq)$         |
|                | VB-tree [20]                           | $O(mn)$          | $O(q + \log q)$ | $O(q + \log q)$ | $O(q + \log q)$ |
|                | PMD [22], MB-tree, EMB-tree [16]       | $O(n)$           | $O(q + \log n)$ | $O(q + \log n)$ | $O(q + \log n)$ |
|                | XB-Tree [15]                           | $O(n)$           | $O(q + \log n)$ | $O(q + \log n)$ | $O(q + \log n)$ |
|                | HAT [17]                               | $O(n)$           | $O(q + \log n)$ | $O(q + \log n)$ | $O(q + \log n)$ |
| Signature-based| Condensed-RSA [18]                     | $O(n)$           | $O(nq)$         | $O(nq)$       | $O(nq)$         |
|                | DSAC [30]                              | $O(mn)$          | $O(mq)$         | $O(mq)$       | $O(mq)$         |
|                | iterative hash [29]                    | $O(n(m + \Omega))$ | $O(q\Omega)$   | $O(q\Omega)$ | $O(q\Omega + m\Omega)$ |
|                | ECC-signature [19]                     | $O(n)$           | $O(q)$          | $O(q)$        | $O(q)$          |
|                | VKD-Tree [24]                          | $O(nmn\Omega)$   | $O(qmn\Omega)$  | $O(qmn\Omega)$ | $O(qmn\Omega)$  |
|                | AuthODS [31]                           | $O(n)$           | $O(q + m\log n)$ | $O(q + m\log n)$ | $O(q + m\log n)$ |
| Accumulation   | AMR [13]                               | $O(mn\log n)$   | $O(mn\log n + q)$ | $O(mn\log n + q)$ | $O(mn\log n + q)$ |
|                | IntegrilDB [25]                        | $O(n)$           | $O(qn\log n + q)$ | $O(qn\log n)$ | $O(qn\log n)$ |
| Interactive    | vSQL [26]                              | $O(n)$           | $O(q)$          | $O(q)$        | $O(q)$          |
| Hardware       | CorrectDB [14]                         | $O(n)$           | $O(q)$          | $O(q)$        | $O(q)$          |
client, the signature-based solutions are more suitable than the tree-based approaches for those queries whose results are of large size. Empirical studies in [19]) also show that the tree-based approaches can take only half time of the signature-based approaches when the query workload is light (less than 10 queries per second). But the performance degrades quickly when the workload increases. When there are data updates, the tree-based approaches incur higher query response time but cheaper verification time than the signature-based approach.

6.2.2 Probabilistic Approaches

The probabilistic approaches have different assumptions of SP. For example, the query-sampling based approach [27] requires that SP to be aware of the verification protocol, while the fake-record based approach [28] relaxes this assumption and makes the verification approach to be transparent to SP; thus SP is not required to be involved in the verification protocol. This reduces the computational overhead by SP. For the query-sampling based approach [27], the verification cost for the client and the client-SP communication overhead grows linearly with the number of queries in the batch. For the fake record based method [28], the cost complexity is linear to the number of fake records. For the other fake record based methods [32], [33], the setup overhead and the verification cost at SP side is linear to the original data size and the number of fake records. To summarize, for both query-sampling and fake-record based approaches, their performance highly relies on the probabilistic integrity guarantee that has to be achieved. Apparently, higher integrity guarantee requires more expensive verification overhead.

6.2.3 Deterministic versus Probabilistic Approaches

The trade-off exists between performance and integrity guarantees. The probabilistic approaches can incur cheaper setup and verification overhead than the deterministic approaches, but with weaker integrity guarantee. According to the empirical study in [28], the fake record based probabilistic approach [28] incurs much cheaper verification setup overhead at the data owner side than the query sampling based approach [27] and the MHT tree based deterministic approach [29]. However, the computational cost at SP side by the fake record based approach [28] can be higher than the two aforementioned approaches, due to the fact that a large number of fake records have to be generated to achieve high probabilistic guarantee.

6.3 Practical Issues

Choice of Cryptographic Primitives for Signature Generation. An effective way to reduce the storage and communication overhead is to generate shorter signatures by using the appropriate cryptographic primitives. As suggested by [19], ECC can replace RSA since it can produce the proof of fewer bits than RSA with comparable security guarantee. For example, an ECC signature that is 160 bits long provides comparable security to a 1024-bit RSA signature [40].

Aggregate Signature versus Hash. The overhead of transmitting and verifying a signature can be very large. One possible approach to reduce the communication overhead (between the client and SP) is to combine the signatures associated with individual entries in VO into one aggregate signature. On the other hand, generating an aggregate signature is much slower than a hash. As shown in [29], verification by using the aggregate signature can be around 100 times slower than hash. Therefore, the choice between aggregate signature and hash should consider these practical concerns.

Priorities of Performance Optimization. As pointed out in [55], the communication between the client and SP (e.g., the cloud) can cost up to 3,500 picocents/bit, 2-3 orders of magnitude higher than the computational cost ($1 \times 10^{-12}$) [14]. Therefore, reducing the communication overhead between the client and SP should be given higher priorities than optimization of computational overhead at the sides of the data owner, SP, and the client. Possible effective ways to reduce the communication overhead is to minimize the VO size or apply a probabilistic authentication method (with high integrity guarantee) instead of a deterministic method.

7 INTEGRITY WITH OTHER SECURITY FEATURES

Other security goals can be considered in parallel with query authentication in the Daas framework. For example, how to efficiently authenticate query results without violating access control policies, and how to verify the integrity of results on privacy-preserving query evaluation. In this section, we review the existing works that integrate query integrity authentication with other security features.

7.1 Integrity and Privacy

Singh et al. [56] consider a two-party scenario, in which a data owner has a private database, and an external querier would like to obtain some information from the private database by sending the queries to the data owner. The external querier desires to obtain a proof that shows the results returned by the database owner indeed reflects the correct evaluation of the submitted query over an un-corrupted version of the database. On the other hand, the data owner does not reveal any private information to the querier except the query results. The goal is to design a query authentication method for private databases, aiming to minimize the amounts of data revealed to the untrusted third party, except the query results. The key idea of [56] to preserve database privacy is to reveal only the hashes of records that are not part of the result. The integrity of query results is verified by reconstructing the root signature of the hash tree, similar to [21].

Wang et al. [57] consider a three-party scenario that contains the data owner, SP, and the client who is not necessarily to be the data owner. To protect the private data from the untrusted SP, the data is encrypted by the methods that support encrypted queries over encrypted databases. To provide query verification guarantee, they design a novel encryption method called dual encryption, whose key idea is to first encrypt the entire dataset \( D \) using a primary encryption key \( k \), and encrypt a selected subset of \( D \) using a secondary encryption key \( k' \). The two encrypted datasets are merged and stored at SP as a single dataset. For each query \( q_i \), the client sends its encrypted query \( q_i'^{k'} \) to SP and gets its result \( \rho(q_i'^{k'}) \). To verify the integrity of \( \rho(q_i'^{k'}) \), the client sends \( q_i^{k'} \) to SP, where \( q_i^{k'} \) and \( q_i'^{k'} \) are semantically identical (i.e.,
they return the same set of records). The client checks if for each replicated record $t \in \rho(q^i_0)$, $t$ also appears in $\rho(q^i_k)$.

### 7.2 Integrity and Access Control

Pang et al. [29] is the first work that considers query authentication with the respect of the access control policy. They consider the completeness verification for selection-projection queries, with the assumption that the original selection-projection query has been re-written as a new selection-projection query. The challenge is to make sure that during the construction of completeness proof, the left and right boundaries of the query results that are used for VO do not violate the access control policy. To address this challenge, the data owner inserts two fictitious entries, a left delimiter $r_0 \in (L, U)$ and a right delimiter $r_{n+1} \in (L, U)$ into $D$. The two fictitious entries ensure that the boundary records used in the proof do not violate the access control policy. For completeness verification, the digital signature of the real records in the query results are constructed. While for the two delimiters, their signatures are constructed by aggregating with the boundary records. Based on the signatures, the completeness verification is performed in the similar way as in [20].

Kundu et al. [58] consider the structured data (such as XML documents) in which both the data structure and content are considered confidential. The access policies specify which nodes and structural information that a user is allowed to access. The goal of [58] is to enable a user to authenticate a subtree $S$ from a tree $T$ without leaking any other nodes from $TS$ (refers to the cut operation). They designed a novel concept called structural signature, which consists of a hash of the structural position, the node content of $x$, and the (salted) signature of the tree. The verification relies on building the correct signature of the tree to show correctness. The hash of the structural position allows the third-party to reconstruct the subtree without access to the original structure.

### 8 NEW RESEARCH DIRECTIONS, OPEN PROBLEMS, AND CHALLENGES

In this survey, we reviewed the existing studies on integrity authentication of SQL query evaluation for the DaaS paradigm. We categorize these works by how the verification is performed and the integrity guarantee that these works can achieve. There exists the trade-off between the integrity guarantee and the performance of the verification methods. The deterministic integrity guarantee can be achieved but with possibly expensive setup cost and verification overhead; such overhead can be improved by relaxing the integrity guarantee to be probabilistic.

In the rest of this section, we discuss several interesting directions for the future work.

### 8.1 Authentication of Query Evaluation on Big Data

The three well-known properties of Big data is 3Vs (volume, variety and velocity). Volume refers to the amount of data, variety refers to the number of types of data and velocity refers to SPeed of data processing. The three properties raise different challenges to the authentication of query evaluation. The big data volume requires that the authentication methods must be scalable; the setup and verification should not add significant overhead on query evaluation itself. The fast velocity requires that the authentication methods must support frequent updates efficiently. The large variety requires that the authentication methods should be able to support various data types. A naive method is to construct an ADS for each data type, and thus a proof for each data type too. But this may bring overwhelming verification overhead. An interesting research direction that worth to explore is how to address the challenges of volume, variety and velocity of Big data for the design of efficient and scalable authentication solutions.

Consider MapReduce, an important computational model for data intensive applications. Intuitively, the MapReduce infrastructure consists of two primitives: (a) a map function that distributes the input to multiple mapper nodes to generate intermediate key-values in parallel, and (b) a reduce function that allows multiple reducer nodes to merge the intermediate pairs associated with the same key and then to generate the final outputs. Intuitively, all mappers and reducers can be compromised and return wrong intermediate and/or final results. There are two main challenges of SQL query evaluation on MapReduce: (1) how to authenticate the query results in the format of key-value pairs; and (2) how to authenticate both the intermediate and final results for MapReduce execution. A few papers have considered integrity verification of MapReduce execution [59], [60], [61]. Intuitively, the intermediate results can be authenticated by applying the existing SQL authentication solutions presented in this survey, e.g., tree-based and signature-based methods, to the output of each mapper. However, this requires to construct an ADS independently for each mapper, which can be extremely expensive. The main challenge is how to construct the ADS to support efficient verification of both intermediate query results by the mappers and final query results by the reducers in the distributed fashion.

### 8.2 Authentication for Various DaaS Outsourcing Paradigms

In general, the outsourcing paradigms can be classified into three types:

- **Infrastructure-as-a-Service (IaaS)** paradigm: the data owner can deploy and run arbitrary software including operating systems and applications. SP provides the computing infrastructure, e.g., data storage and hardware for the computation. A typical IaaS example is Amazon EC2 Web Service [6].
- **Platform-as-a-Service (PaaS)** paradigm: SP delivers a computing platform, typically including operating system, execution environment, database, and web server. Some well-known PaaS offers include Microsoft Azure [10] and Google Cloud SQL [8].
- **Software-as-a-Service (SaaS)** paradigm: The clients use SP’s applications running on an outsourced infrastructure (e.g., the cloud). The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface.

Different computational paradigms bring different challenges of authentication. Most of the existing authentication methods do not consider the SP of Paas and SaaS paradigms.
that has more cheating power on the computations than the IaaS paradigm, as the client has neither knowledge of the computations nor how these computations are configured and executed by the SP. It remains open to design authentication methods that can deal with black-box attacks.

8.3 Adapting Authentication Techniques to Other Types of Outsourced Computations

In practice, the outsourcing paradigm supports a broader range of data analytics services, ranging from simple aggregation to Web search to sophisticated data mining and learning, to name a few. While this survey focuses on authentication of SQL query evaluation only, whether the existing SQL authentication methods are adaptable to other types of outsourced computations remains open. Below we present a few examples of outsourced computations, and discuss the challenges and potential research directions by adapting the existing SQL authentication techniques to these types of outsourced computations.

8.3.1 Keyword Search on Web

Web search engines are typical outsourced computations. The search service providers may return incorrect search results due to various reasons (e.g., to make profits from advertisements) [62]. The SP may also return the results in a wrong ranking (e.g., to promote some websites). Therefore, the authentication goals are two-fold: (1) to verify that the returned keyword search results are sound, complete, and fresh, and (2) to verify the ranking of the returned results is accurate.

Existing Works. [62], [63] have initiated the research on authentication of soundness, completeness, and freshness of the Web-search content. They demonstrated that bilinear pairing and MHT can be used to authenticate the keyword search results.

Open Problems and Challenges. How to verify the ranking of the results, especially the top-k results, remains unsolved. It is interesting to explore if the existing top-k query authentication methods under different contexts (e.g., sensor networks [64] and location-based services [65]) can be adapted to keyword search queries.

8.3.2 Outsourced Data Mining and Machine Learning

Big corporations like Amazon, Google and Microsoft are providing cloud-based data mining and learning services in various forms.

Existing Works. The existing authentication solutions include various types of data mining computations, including association rule mining [66], [67], [68], outlier mining [69], [70], clustering [71], Bayesian network construction [72], and collaborative filtering [73], [74]. Many of these solutions (e.g., [70], [71], [72]) consider the probabilistic approaches.

Open Problems and Challenges. There are several interesting open research questions. First, can any existing ADS structure (e.g., MB-tree [16] and VB-tree [20]) for SQL query authentication be adapted to the authentication solutions for outsourced machine learning and data mining computations? Second, given the high complexity of machine learning computations, the deterministic approach may not be appropriate as it can bring expensive overhead. It is not straightforward how to adapt the existing probabilistic SQL query verification approaches (e.g., by inserting fake records [28]) to the outsourced machine learning computations, as inserting fake records can change, perhaps unavoidably, the machine learning results (e.g., classification and clustering) of the original data. The challenge is how to design those fake records for authentication while keeping the original data analytics results unchanged. Furthermore, these probabilistic authentication methods may be vulnerable against the attacker who is aware of the authentication mechanism, for example, how the fake records are constructed. An interesting research direction is to design robust probabilistic verification methods that address the trade-off between efficiency and the robustness of the authentication methods.

8.3.3 Spatial Query Evaluation

The goal of authentication of spatial query evaluation is to verify if the returned locations satisfy the given constraints (e.g., within a range, one of the nearest neighbor, etc.).

Existing Works. A large body of work has considered authentication of query evaluation on spatial databases (e.g., [75], [76], [77], [78], [79]). These works share the same idea of the ADS-based authentication methods for SQL queries, and design various ADS techniques (e.g., Voronoi diagram and R-trees) to represent the underlying spatial database.

Open Problems and Challenges. The main challenge is how to compute the distances efficiently for verification. In particular, how to design efficient, online authentication methods that can deal with dynamic location data that are updated frequently remains unsolved. This requires that the authentication methods can support fast updates on the ADS, quick construction of the proof of the query results, and cheap verification at the client side who may use resource-constrained devices such as mobile phones for verification.

ACKNOWLEDGMENTS

This material is based upon work supported by the US National Science Foundation (NSF) under Grant No. 1350324 and 1464800.

REFERENCES

[1] D. E. Weisman and C. Buss, “Database functionality high, analytics lags,” in Forrester Brief: Bus. Technographics North America, 2001.
[2] D. J. Abadi, “Data management in the cloud: Limitations and opportunities,” IEEE Data Eng. Bull., vol. 32, no. 1, pp. 3–12, 2009.
[3] T. Claburn, “Yahoo scales its web analytics database to petabyte range.” Accessed: Oct. 22, 2019. [Online]. Available: http://www.dbms2.com/2008/05/29/yahoo-scales-web-analytics-database-petabyte/
[4] C. Monash, “The 1-petabyte barrier is crumbling.” Accessed: Oct. 22, 2019. [Online]. Available: http://www.networkworld.com/community/node/31439
[5] H. Hacigumus, B. Iyer, and S. Mehrotra, “Providing database as a service,” in Proc. 18th Int. Conf. Data Eng., 2002, pp. 29–38.
[6] Amazon elastic compute cloud (EC2). Accessed: Oct. 22, 2019. [Online]. Available: http://aws.amazon.com/ec2/.
[7] Amazon simple storage service (S3). Accessed: Oct. 22, 2019. [Online]. Available: http://aws.amazon.com/s3/.
[8] Google cloud SQL. Accessed: Oct. 22, 2019. [Online]. Available: https://cloud.google.com/sql/
[9] Microsoft cloud database service. Accessed: Oct. 22, 2019. [Online]. Available: https://azure.microsoft.com/en-us/services/sql-database/
[10] Windows azure cloud. Accessed: Oct. 22, 2019. [Online]. Available: http://www.windowsazure.com/en-us/
[62] M. T. Goodrich, C. Papamanthou, D. Nguyen, R. Tamassia, C. V. Lopes, O. Ohrimenko, and N. Triandopoulos, “Efficient verification of web-content searching through authenticated web crawlers,” Proc. Very Large Data Base Endowment, vol. 5, no. 10, pp. 920–931, 2012.

[63] D. Nguyen, “Verification of web-content searching through authenticated web crawler,” Brown University, Providence, RI, 2011.

[64] R. Zhang, J. Shi, Y. Liu, and Y. Zhang, “Verifiable fine-grained top-k queries in tiered sensor networks,” in Proc. IEEE INFOCOM Conf., 2010, pp. 1–9.

[65] Q. Chen, H. Hu, and J. Xu, “Authenticating top-k queries in location-based services with confidentiality,” Proc. VLDB Endowment, vol. 7, no. 1, pp. 49–60, 2013.

[66] B. Dong, R. Liu, and H. W. Wang, “Result integrity verification of outsourced frequent itemset mining,” in Proc. IFIP Annu. Conf. Data Appl. Security Privacy, 2013, pp. 258–265.

[67] B. Dong, R. Liu, and H. W. Wang, “Trust-but-verify: Verifying result correctness of outsourced frequent itemset mining in data-mining-as-a-service paradigm,” IEEE Trans. Services Comput., vol. 9, no. 1, pp. 18–32, Jan./Feb. 2016.

[68] B. Dong, R. Liu, and H. W. Wang, “Integrity verification of outsourced frequent itemset mining with deterministic guarantee,” in Proc. IEEE Int. Conf. Data Mining, 2013, pp. 1025–1030.

[69] B. Dong, H. W. Wang, A. Monreale, D. Pedreschi, F. Giannotti, and W. Guo, “Authenticated outlier mining for outsourced databases,” IEEE Trans. Depend. Secure Comput., vol. 21, no. 9, pp. 1545–1558, 2017.

[70] R. Liu, H. W. Wang, A. Monreale, D. Pedreschi, F. Giannotti, and W. Guo, “AUDIO: An integrity auditing framework of outlier-mining-as-a-service systems,” in Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases, 2012, pp. 1–18.

[71] R. Liu, H. Wang, P. Mordohai, and H. Xiong, “Integrity verification of K-means clustering outsourced to infrastructure as a service (IaaS) providers,” in Proc. SIAM Int. Conf. Data Mining, 2013, pp. 632–640.

[72] R. Liu, H. Wang, and C. Yuan, “Result integrity verification of outsourced Bayesian network structure learning,” in Proc. SIAM Int. Conf. Data Mining, 2014, pp. 713–721.

[73] J. Vaidya, I. Yakut, and A. Basu, “Efficient integrity verification for outsourced collaborative filtering,” in Proc. Int. Conf. Data Mining, 2014, pp. 560–569.

[74] Q. Tang, B. Pejo, and H. Wang, “Protect both integrity and confidentiality in outsourcing collaborative filtering computations,” in Proc. Int. Conf. Cloud Comput., 2016, pp. 941–946.

[75] Y. Jing, L. Hu, W.-S. Ku, and C. Shahabi, “Authentication of k nearest neighbor query on road networks,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 6, pp. 1494–1506, Jun. 2014.

[76] S. Papadopoulos, L. Wang, Y. Yang, D. Papadias, and P. Karras, “Authenticated multistep nearest neighbor search,” IEEE Trans. Knowl. Data Eng., vol. 23, no. 5, pp. 641–654, May 2011.

[77] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi, “Spatial query integrity with voronoi neighbors,” IEEE Trans. Knowl. Data Eng., vol. 25, no. 4, pp. 863–876, Apr. 2013.

[78] H. Hu, Q. Chen, and J. Xu, “VERDICT: Privacy-preserving authentication of range queries in location-based services,” in Proc. IEEE 29th Int. Conf. Data Eng., 2013, pp. 1312–1315.

[79] D. Yung, E. Lo, and M. L. Yu, “Authentication of moving range queries,” in Proc. 21st ACM Int. Conf. Inf. Knowl. Manage., 2012, pp. 1372–1381.

**Bo Zhang** received the BS degree from the Wuhan University of Technology of China, Hubei, China. Currently, he has been working toward the PhD degree at the Stevens Institute of Technology, Hoboken, New Jersey since 2015. His research interests include database, verifiable computation, and data mining.

**Boxiang Dong** received the PhD degree in computer science from the Stevens Institute of Technology, New Jersey. He is an assistant professor with the Computer Science Department, Montclair State University, New Jersey. He is dedicated to facilitating the integration of big data analysis and cybersecurity. His research interests include verifiable computing, data mining, anomaly detection, data security, and privacy.

**Wendy Hui Wang** received the PhD degree in computer science from the University of British Columbia, Vancouver, Canada. She is an associate professor with the Computer Science Department, Stevens Institute of Technology, New Jersey. Her research interests include data management, data mining, database security, and data privacy. She is a member of the editorial boards of the *Journal of Information Technology and Architecture* and the *Journal of Information Systems*.

---

**For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/csdl](http://www.computer.org/csdl).**