Database Management as a Service: Challenges and Opportunities

Divyakant Agrawal [#1], Amr El Abbadi [#2], Fatih Emekci [*3] Ahmed Metwally [@4]

[#] *Department of Computer Science, University of California at Santa Barbara*
Santa Barbara, CA 93106, USA
[1] agrawal@cs.ucsb.edu
[2] amr@cs.ucsb.edu

[*] *LinkedIn Corporation*
2029 Stierlin Court, Mountain View, CA 94043, USA
[3] fatihemekci@gmail.com

[@] *Google Inc.*
1600 Amphitheatre Parkway, Mountain View, CA 94043, USA
[4] ametwally@gmail.com

***Abstract*— Data outsourcing or database as a service is a new paradigm for data management in which a third party service provider hosts a database as a service. The service provides data management for its customers and thus obviates the need for the service user to purchase expensive hardware and software, deal with software upgrades and hire professionals for administrative and maintenance tasks. Since using an external database service promises reliable data storage at a low cost it is very attractive for companies. Such a service would also provide universal access, through the Internet to private data stored at reliable and secure sites. However, recent governmental legislations, competition among companies, and database thefts mandate companies to use secure and privacy preserving data management techniques. The data provider, therefore, needs to guarantee that the data is secure, be able to execute queries on the data, and the results of the queries must also be secure and not visible to the data provider. Current research has been focused only on how to index and query encrypted data. However, querying encrypted data is computationally very expensive. *Providing an efficient trust mechanism* to push both database service providers and clients to behave honestly has emerged as one of the most important problem before data outsourcing to become a viable paradigm. In this paper, we describe scalable privacy preserving algorithms for data outsourcing. Instead of encryption, which is computationally expensive, we use distribution on multiple data provider sites and information theoretically proven secret sharing algorithms as the basis for privacy preserving outsourcing. The technical contributions of this paper is the establishment and development of a framework for efficient fault-tolerant scalable and theoretically secure privacy preserving data outsourcing that supports a diversity of database operations executed on different types of data, which can even leverage publicly available data sets.**

I. Introduction

Internet-scale computing has resulted in dramatic changes in the design and deployment of information technology infrastructure components. Cloud computing has been gaining in popularity in the commercial world, where various computing based capabilities are provided as a service to clients, thus relieving those clients from the need to develop expertise in these capabilities, as well as the need to manage and maintain the software providing these services. Amazon, for example, has created the service EC2, which provides clients with scalable servers; as well as another service S3, which provides scalable storage to clients. Recently, NSF partnered with Google and IBM to offer academic institutions access to large scale distributed infrastructure under the NSF CLuE program. There has clearly been a radical paradigm shift due to the wide acceptance of and reliance on Internet and Web-based technologies.

One of the reasons for the success of Internet-scale computing is the role it has played in eliminating the size of an enterprise as a critical factor in its economic success. An excellent example of this change is the notion of *data centers* which provide clients with the physical infrastructure needed to host their computer systems, including redundant power supplies, high bandwidth communication capabilities, environment monitoring, and security services. Data centers eliminate the need for small companies to make a large capital expenditure in building an infrastructure to create a global customer base. The data center model has been effective since it allows an enterprise of any size to manage growth with the popularity of its product or service while at the same time also allows the enterprise to cut its losses if the launched product or service does not succeed. During the past few years we have seen a rapid acceleration of innovation in new business paradigms and data centers have played a very important role in this process.

In addition to the physical infrastructure needed to support Internet and web-based applications, such applications have data management needs as well. To enable more sophisticated business analysis and user customization, e-commerce applications maintain data or log information for every user interaction rather than only storing transaction data (e.g. sales transactions in the retail industry). This trend has resulted in an explosive growth in the amount of data associated with these applications. Storage and retrieval of such data poses monumental challenges especially for relatively small-sized companies since the cost of data management is estimated to be five-to-ten times higher than the data acquisition cost [1]. More importantly, in-house data management requires a much higher level skill set to deal with issues of storage technologies, capacity planning, fault-tolerance and reliability, disaster recovery, and DBMS and Operating Systems software upgrades. Most commercial entities would rather direct their valuable technical resources and engineering talent to focus on their business applications instead of becoming full-time data management companies.

Due to the above concerns, *data outsourcing* or *database as a service* has emerged as a new paradigm for data management in which a third party service provider hosts a database and provides the associated software and hardware support. The Database Service Provider (DBSP) provides data management for its customers, and thus obviates the need for the customer to purchase expensive hardware and software, deals with software upgrades, and hires professionals for administrative and maintenance tasks. Since using a DBSP promises reliable data storage at a low cost, it is very attractive for large enterprises such as commercial entities, intelligence agencies, and other public and private organizations. Such a service would also provide universal access through the Internet to private data stored at reliable and secure sites. A client company can outsource their data, and its employees then have access to the company data irrespective of their current locations. Rather than carrying data with them as they travel or logging remotely into their home machines, employees can store and access company data with the DBSP, thus eliminating the risk of data lost or theft.

Although the data outsourcing paradigm has compelling economic and technical advantages, its adaptation has not mirrored the success of data centers. The main reason for this failure is that recent governmental legislations, competition among companies, and database thefts mandate that enterprises use secure and privacy preserving data management techniques. Using an external database service is an example of a straightforward client-server application in an environment where service providers and clients are honest and clients do not hesitate to share their data with database service providers. However, this is usually not the case, and thus the research challenge is to build a robust and efficient service to manage data in a secure and privacy-preserving manner. The DBSP, therefore, needs to guarantee that data is secure, that the results of the queries must also be secure, and that the queries can be executed efficiently and correctly,

Previous research in the context of secure data outsourcing has focused on these areas independently. In the case of ensuring *security* or equivalently *confidentiality* of outsourced data, most of the research is concerned with ensuring that even though the data is outsourced to a third party, the individual data values should not be discernible to the service provider [1], [2], [3], [4], [5], [6], [7], [8], [9]. In the case of the security of query results, also called *private information retrieval*, the problem has been studied as the theoretical formulation where the client must be able to retrieve the i^{th} element from N data elements without the service provider discovering that the client is interested in the i^{th} element [10], [11], [12], [13], [14], [15], [16]. Finally, in the last case we are concerned with a malicious environment, and therefore need to ensure completeness and correctness of user queries in that the results returned by the service provider are indeed the exact answers to the user queries [17], [18], [19], [20], [21]. If, for example, the service provider corrupted the data, it would be impossible to recover it for the service user. To be able to use external DBSPs in real-world settings, there must be a mechanism to recover the data and also to verify that data has been corrupted. *Providing a trust mechanism* to ensure both DBSPs and clients behave honestly has emerged as one of the most important problem that must be overcome before data outsourcing becomes a viable paradigm. Clearly, a wide adaptation of data outsourcing framework will only be possible if all three issues are adequately addressed.

Technical Rationale

Most approaches proposed for secure and private data outsourcing are based on data encryption [22]. In the recent International Conference on Very Large Data bases (VLDB'2007), Sion [22] presented a comprehensive review of the current state-of-the-art in secure data outsourcing. A review of the tutorial notes reveals that indeed most of the techniques primarily rely on advanced cryptographic algorithms and more recently there is some effort to design and develop special purpose hardware technology to overcome the overhead associated with data encryption. In fact, in his concluding slide, Sion states that the practical maturity of secure data outsourcing is in its *infancy* and adaptation of such technology is *barely crawling*. Secure data outsourcing is complementary to the notion of data centers in that it will enable enterprises to outsource both application processing as well as data management to external entities and thus leveraging from economies of scale resulting in significant efficiencies in the Information Technology infrastructures.

II. BACKGROUND

A. *Encryption-based Data Security*

Current research on data security use encryption to hide the content of the data from service providers [1], [3], [5], [7], [8], [9]. The concept of *database as a service* has been of interest to the research community under various guises. NetDB2 [1] was developed and deployed as a database service on the Internet. NetDB2 directly addresses two of the main challenges in developing databases as a service, namely data privacy and performance. NetDB2 uses encryption to ensure privacy and is the first work that directly addresses and evaluates the issue of performance, which is critical for success in databases. Recently [23] proposed using homomorphic encryption to support secure aggregate outsourcing. In [19] Sion introduced the notion of query execution assurance in outsourced databases, namely, assurances are provided by the database server that the served client queries were in fact correctly executed on the entire outsourced database. In [24], the vulnerabilities of using a single key for data encryption is raised and hence the authors [24] propose using a hierarchical key assignment scheme. Anciaux et al. [25] address the interesting problem of executing privacy preserving operations on both public as well as private data, when the latter are stored on smart USB keys.

The computational complexity of encrypting and decrypting data to execute a query increases the query response time. Therefore, this complexity is one of the bottlenecks in current solutions [26]. In fact, Agrawal et al. [26] show that computing a privacy preserving intersection problem using encryption results in a very high time complexity. The cost estimation of encryption based approach on a synthetic data consisting of 10 documents at one site and 100 documents at another site (each with 1000 words) could take as much as 2 hours of computation and approximately 3 Gigabits of data transmission. Similarly, for a real dataset consisting of approximately 1 million medical records, the encryption based approach takes approximately 4 hours of computation time and and 8 Gbits of data transmission. Another problem with the data encryption approach is that finding the required tuples to execute the query over encrypted data is a significant challenge. In order to solve this problem, current proposals reveal some information about the content of the data to be used in filtering the required tuples [1], [2]. With a good filtration mechanism, the communication cost of retrieving data from service providers would be less and thus the query response time would be much better. However, the quality of the filtration process strictly depends on the amount of information revealed to the service provider. Therefore, there is a privacy performance tradeoff in these solutions. Finally, in order to execute range queries efficiently order preserving data encryption mechanisms have been proposed [3]. However, it has also been argued that order preservation may weaken data security [5].

B. Private Information Retrieval

Interestingly, the problem of private information retrieval has been a research topic for more than a decade. The problem was first proposed in the context of a user accessing third-party data without revealing to the third-party his/her exact interests [11]. An example scenario is an analyst who wants to retrieve information from an investor database made available by a certain company, but who would not like his/her future intentions be exposed to that company. Although the original formulation of this problem was with respect to third-party data it is very much applicable in the context of outsourced data. In particular, users may not want to reveal their queries to the service provider since this information can compromise the privacy of their behavior patterns.

Formally, private information retrieval is stated as follows: the database is modeled as a string x of length N held at a remote server, and the user wants to retrieve the bit x_i for some i, without disclosing any information about i to the server [11]. A trivial solution would be for the user to retrieve the entire database. A simple proof establishes that if we only have one server, the trivial solution is the best we can hope for [11]. A way to obtain sub-linear communication complexity is to replicate the database at several servers. It has been shown that with k servers the communication complexity can be reduced to $O(N^{\frac{1}{2k-1}})$. There is a long history of theoretical research in this area and most solutions rely on the availability of multiple servers or multiple service providers [10], [11], [12], [13], [14], [15], [16].

The notion of private information retrieval which ensures privacy of user queries has also been extended to the case where the privacy of data is a concern. This is referred to as *symmetric private information retrieval* [27], [28], [29]. In a recent work by Sion and Carburnar [16], the authors have challenged the computational practicality of both private information retrieval and symmetric private information retrieval. In that, the authors through extensive experimentation and evaluation have established that the private information retrieval protocols are several orders of magnitude slower than the trivial protocol of transferring entire database to the client to ensure user privacy. Thus, in essence the authors doubt the viability of private information protocols in practice and state that alternative approaches are warranted to address the very *real* and *practical* problem of data outsourcing.

C. Information Distribution

In the area of computer and data security, there is an orthogonal approach which is based on information dispersal or distribution instead of encryption. Most of this work on security arose in the context of communicating a secret value from one party to another. Many approaches rely on encrypting the secret value using encryption keys and ensure that the information can only be revealed to a party that has the key. Shamir [30] proposed an orthogonal scheme that does not rely on the notion of keys. Instead, he proposed splitting the secret into n pieces such that the secret value can only be revealed if one has access to any k of these n pieces, $k < n$. This scheme is shown to be information-theoretically secure as long as there is some guarantee that the adversary cannot access k pieces. Unlike encryption methods, Shamir's secret sharing algorithm is computationally efficient. Our intent is take a distributed approach towards secure data outsourcing in that we want to explore using secret-sharing approach and multiple service providers. The advantage of this approach is that it addresses both data security as well as privacy-preserving querying of outsourced data.

In [31], [32], we proposed using Shamir's secret sharing algorithm to execute privacy preserving operations among a set of distributed data warehouses. In [31], a middleware, Abacus, was developed to support selection, intersection and join operations. This approach was designed for data warehouses and took advantage of the dimension table in the star scheme to hide information using inexpensive one-way hash functions. In [32], this work was generalized to any type of database, and used distributed third parties and Shamir's secret sharing algorithm to secure information and support privacy preserving selection, intersection, join and aggregation operations (such as SUM and MIN/MAX operations) on a set of distributed databases.

III. Overview of the Proposed Approach

In this section, we formulate the secure data outsourcing problem and then briefly discuss our proposed approach towards an effective solution. Our approach is to use data dispersion on multiple servers instead of data encryption to achieve data security and privacy of user queries. This approach is not only efficient, but also exploits the paradigm of Internet-scale computing by taking advantage of the large number of available resources. Assume that a client, which we refer to as the *data source* D, wants to outsource its data to eliminate its database maintenance cost by using the database service provided by database service providers $DAS_1, ..., DAS_n$. D needs to store and access its data remotely without revealing the content of the database to any of the database services. For the sake of this discussion, assume D has a single table $Employees$, with employee names and salaries, in its database and stores $Employees$ using the services provided by $DAS_1, ..., DAS_n$. After storing $Employees$, D needs to query $Employees$ without revealing any information about either the content of the table or queries. D can pose any of the following queries over time:

1) Exact match queries such as: *Retrieve all information about employees whose name is 'John'*.
2) Range queries such as: *Retrieve all information about employees whose salary is between 10K and 40K*.
3) Aggregate queries such as MIN/MAX, MEDIAN, SUM and AVERAGE. Aggregates can be over ranges for example *sum of the salaries of employees whose salary is between 10K and 40K*. In addition, they can be over exact matches such as *average of the salaries of all employees whose name is 'John'*.

We propose a complete approach to execute exact match, range, and aggregation queries in a privacy preserving manner. The goal is to build a practical scalable system and answer queries without revealing any information. Throughout the paper, we will assume that there are two kinds of attributes in tables namely *numeric attributes* such as salary and *non-numeric attributes* such as name. We will contrast the work in [1], [2] referred to as *data encryption*, wherever appropriate, with our proposed technique so as to highlight the differences and compare them.

In our solution, data is divided into n shares and each share is stored in a different service provider. When a query is generated at a data source, it is rewritten, the relevant shares are retrieved from the service providers, and the query answer is reconstructed at the data source. In order to answer queries, any k of the service providers must be available. The main idea here is that the service providers are not able to infer anything about the content of the data they store, and still the data source is able to query its database by incurring reasonable communication and computation costs.

We start by discussing simple techniques of outsourcing a singleton numeric attribute such as salary in an idealized environment. Then, we generalize our techniques to deal with more complex operations, and propose extensions for non-numeric data. The solution is based on Shamir's secret sharing method [30]. Data source D divides the numeric value v_s into n shares and stores one share at each service provider, DAS_1, DAS_2, ..., DAS_n, such that knowledge of any k ($k \leq n$) shares in addition to some secret information, X, known only to the data source is required to reconstruct the secret. Since, even complete knowledge of $k-1$ service providers cannot reveal any information about the secret even if X is known, this method is information theoretically secure [30].

In the secret sharing method, data source D chooses a random polynomial $q(x)$ of degree $k-1$ where the constant term is the secret value, v_s, and secret information X which is a set of n random points, each corresponding to one of the database service providers. Then, data source D computes the share of each service provider as $q(x_i)$, $x_i \in X$ and sends it to database service provider DAS_i. To reconstruct the secret value v_s, the data source retrieves shares from the service providers. The shares can be rewritten as follows:

$$\begin{aligned} shares(v_s, 1) &= q(x_1) = ax_1^{k-1} + bx_1^{k-2} ... + v_s \\ shares(v_s, 2) &= q(x_2) = ax_2^{k-1} + bx_2^{k-2} ... + v_s \\ &\vdots \\ shares(v_s, n) &= q(x_n) = ax_n^{k-1} + bx_n^{k-2} ... + v_s \end{aligned}$$

The secret value can be reconstructed using any k of the above equations since there are k unknowns including the secret value v_s. In order to reconstruct the secret value v_s, any set of k service providers will need to share the information they have received, and they need to know the set of secret points, X, used by D. Since only data source D knows X, only it can reconstruct the secret after getting at least k shares from any k of the service providers.

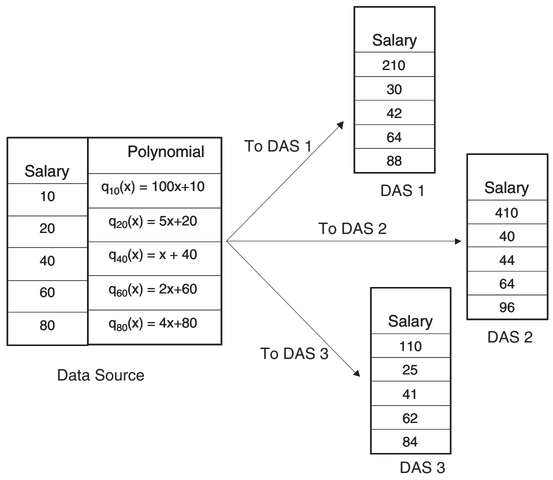

Fig. 1. Demonstration of Example 1

We illustrate the secret sharing model using a concrete example. Assume that data source D needs to outsource the salary attribute of an $Employees$ table using 3 database service providers, DAS_1, DAS_2 and DAS_3. In order to do this, it chooses 5 random polynomials with degree one for each salary value in the table whose constant term is the salary ($n = 3$ and $k = 2$). In addition, secret information X, $X = \{x_1 = 2, x_2 = 4, x_3 = 1\}$, is also chosen one for each database service provider. Therefore, the polynomials would be $q_{10}(x) = 100x + 10$, $q_{20}(x) = 5x + 20$, $q_{40}(x) = x + 40$, $q_{60}(x) = 2x + 60$ and $q_{80}(x) = 4x + 80$ for salaries $\{10, 20, 40, 60, 80\}$ respectively. Then, it sends $\{q_{10}(x_i), q_{20}(x_i), q_{40}(x_i), q_{60}(x_i), q_{80}(x_i)\}$ to service provider DAS_i to store them. The data outsourcing step is illustrated in Figure 1. Note that neither the polynomials nor the salaries are stored at the service provider and Figure 1 shows the information stored at each of the service provider. When a query is initiated at the data source, it needs to retrieve the shares corresponding to all salaries from the service providers, i.e., $\{q_{10}(x_i), q_{20}(x_i), q_{40}(x_i), q_{60}(x_i), q_{80}(x_i)\}$ from DAS_i. After this, it needs to find out the coefficient of each polynomial q and thus all secret salaries (note that receiving any k shares is enough for this since the polynomials are of degree $k-1$). In our example, data source D needs to receive shares from any 2 of the service providers and uses the set of secret values X, to compute the coefficients of polynomials $q_{10}, q_{20}, q_{40}, q_{60}$ and q_{80} and thus salaries, $10, 20, 40, 60$ and 80 can be retrieved.

IV. Practical Solutions for Secure Data Outsourcing

In this section, we extend the techniques developed in the previous section to only retrieve the required data from service providers or a small superset and thus reduce the computation and communication cost in query processing. In the simple solution described above, the database service providers are primarily used as storage servers. They do not play any role in the query processing itself and therefore the proposed approach is not practical. This will result in a large communication cost since the entire database needs to be retrieved from the service provider for every query. Furthermore, the data source itself will become a processing bottleneck since it needs to process all user queries. in fact, this is exactly the same as the same overhead as in the case of data encryption with a single server. In the rest of the paper we use this idealized solution as a starting point and propose refinements to make the solution more practical and scalable.

We now propose an extension to the information dispersal method, which will allow the retrieval of only the required tuples from the service providers instead of a superset. The key observation to achieve this is that the order of the values in the domain $DOM = \{v_1, v_2, ..., v_N\}$ needs to remain the same in the shares of the service providers. In other words, if data source D needs to outsource secret values from domain DOM and $v_1 < v_2 < ... < v_N$, the shares of a service provider DAS_i, $share(v_1, i), share(v_2, i), .., share(v_n, i)$, derived from $v_1, v_2, .., v_N$ respectively need to preserve the order (i.e., $share(v_1, i) < share(v_2, i) < .. < share(v_N, i)$). Since the order of the shares at the service provider is not preserved in the solution in Section 3, database service providers cannot filter the data. However, If we had a mechanism to construct the polynomials calculating shares in an order preserving manner for a specific domain, then data source D could retrieve only the required tuples instead of a superset to answer a query. We now propose an order preserving polynomial building technique to achieve this goal. Without loss of generality, we will assume that polynomials are of degree 3 and in the following form $ax^3 + bx^2 + cx + d$ (i.e., $k = 4$). Given any two secret values v_1 and v_2 from a domain DOM, we need to construct two polynomials $p_{v_1}(x) = a_1x^3 + b_1x^2 + c_1x + v_1$ and $p_{v_2}(x) = a_2x^3 + b_2x^2 + c_2x + v_2$ for these values such that $p_{v_1}(x) < p_{v_2}(x)$ for all x points if $v_1 < v_2$. The key observation for our solution is that $p_{v_1}(x) < p_{v_2}(x)$ for all positive x values if $a_1 < a_2$, $b_1 < b_2$, $c_1 < c_2$ and $v_1 < v_2$. We first present a straightforward approach to construct a set of order preserving polynomials and show why it is not secure. Then, we present a secure method for constructing such order preserving polynomials.

A straightforward method to form a set of order preserving polynomials for a specific domain is to use monotonically increasing functions of the secret values to determine the coefficients of the polynomials. In this scheme, we need three monotonically increasing functions f_a, f_b and f_c to find the coefficients of the polynomial $p_{v_s} = ax^3 + bx^2 + cx + v_s$ which is used to divide the secret value v_s. The coefficients of the polynomial p_{v_s} are the values of the monotonically increasing functions of the secret value v_s where $a = f_a(v_s), b = f_b(v_s)$ and $c = f_c(v_s)$. Therefore, for two secret values v_1 and v_2 ($v_1 < v_2$) and their respective polynomials $p_{v_1}(x) = f_a(v_1)x^3 + f_b(v_1)x^2 + f_c(v_1)x + v_1$ and $p_{v_2}(x) = f_a(v_2)x^3 + f_b(v_2)x^2 + f_c(v_2)x + v_2$, the value of $p_{v_1}(x)$ is always less than the value of polynomial $p_{v_2}(x)$ for all x values. Since any service provider DAS_i gets the value of the polynomials at point x_i, the share coming from secret value v_1, $share(v_1, i)$ would always be less than the share coming from the secret value v_2, $share(v_2, i)$ (i.e., $p_1(x_i) < p_2(x_i)$). However, this solution is not secure enough to hide secret values from the service providers. For example, assume the following monotonic functions are used: $f_a(v_s) = 3v_s + 10$, $f_b(v_s) = v_s + 27$ and $f_c(v_s) = 5v_s + 1$. Then, the share of data source DAS_i from secret value v_1 would be $p_1(x_i) = (3v_1 + 10)x_i^3 + (v_1 + 27)x_i^2 + (5v_1 + 1)x_i + v_1$ which is $p_1(x_i) = (3x_i^3 + x_i^2 + 5x_i + 1)v_1 + (10x_i^3 + 27x_i^2 + x_i)$. Basically, the secrets are multiplied by the same constants and then the same constant is added to compute the share of a service provider for all secret values. Therefore, if a service provider is able to break this method for one secret item can determine the complete set of the secret values.

Since the above approach to construct an order preserving polynomial is not secure, we propose another scheme to build order preserving polynomials for values from a specific domain. In particular, we propose a secure method using different coefficients for each secret value so that service providers cannot know the relation between secret values except the order. In polynomial construction, the coefficients a, b and c are chosen from the domains DOM_a, DOM_b and DOM_c. Since the coefficients can be real numbers, the sizes of the coefficient domains are independent from the data domain size. For finite domain $DOM = \{v_1, v_2, ...v_N\}$, the domains DOM_a, DOM_b and DOM_c are divided into N equal sections. For example DOM_a is divided into N slots : $[1, \frac{|Dom_a|}{N}]$ for v_1, $[\frac{|Dom_a|}{N} + 1, 2\frac{|Dom_a|}{N}]$ for v_2,...,$[(N-1)\frac{|Dom_a|}{N} + 1, |Dom_a|]$ for v_N. After this, coefficient a_{v_i} for value v_i is selected from the slot $[(i-1)\frac{|Dom_a|}{N} + 1, i\frac{|Dom_a|}{N}]$ with the help of hash function h_a which maps v_i to a value from $[(i-1)\frac{|Dom_a|}{N} + 1, i\frac{|Dom_a|}{N}]$. The other coefficients b_{v_i} and c_{v_i} are computed similarly with the hash functions from domains Dom_b and Dom_c. Finally, the polynomial used to divide the secret value v_i into shares would be $p_{v_i}(x) = a_{v_i}x^3 + b_{v_i}x^2 + c_{v_i}x + v_i$.

We now consider the security of the proposed polynomial construction technique. Basically, we discuss what a service provider can infer from the stored data and then show that it cannot know the content of the data with the inferred information. From the stored data, service provider DAS_i can know an upper bound on the sum of the domain sizes (i.e., $|DOM| + |Dom_a| + |Dom_b| + |Dom_c|$). This can only happen when it stores the last secret value from DOM and the coefficients are mapped to the last slots of the domains for the last secret value v_N in the domain. Assume this worst case. Then, the polynomial for secret value v_N would be $P_{v_N}(x) = |Dom_a|x^3 + |Dom_b|x^2 + |Dom_c|x + v_N$ and the share of DAS_i would be $share(i, v_N) = P_{v_N}(x_i) = |Dom_a|x_i^3 + |Dom_b|x_i^2 + |Dom_c|x_i + v_N$. From this share, DAS_i can only know an upper bound on the sum of the sizes of the domains and that upper bound is too loose to infer something about the content of the data. Therefore, we claim that the database service providers can only know an upper bound on the sum of the domains from the stored information. Furthermore, database service provider DAS_i cannot know each domain size or the exact value of the sum of the coefficient domain sizes even if it knows the secret point x_i, in the worst case scenario described above. Because, there are four unknowns, Dom_a, Dom_b, Dom_c and v_N, in the share of DAS_i, $share(i, v_N) = P_{v_N}(x_i) = |Dom_a|x_i^3 + |Dom_b|x_i^2 + |Dom_c|x_i + v_N$ (assuming x_i is known). Thus, these unknowns cannot be found.

In addition to the security guarantees, for two secret values v_i and v_j from the same domain, data source DAS_i will get its shares $share(v_i, i) = p_{v_i}(x_i)$ (share of DAS_i from v_i) and $share(v_j, i) = p_{v_j}(x_j)$. If $v_i < v_j$ then $share(v_i, i) < share(v_j, i)$ due to the polynomial construction method. That is, for any two secret values v_i and v_j from the same domain, the shares of data source DAS_i, $share(v_i, i) = p_{v_i}(x_i)$ (share of DAS_i from v_i) and $share(v_j, i) = p_{v_j}(x_j)$, preserves the order (i.e., if $v_i < v_j$ then $share(v_i, i) < share(v_j, i)$).

V. Research Directions & Challenges

A. Query Processing

In this section, we briefly discuss how different queries are processed in the secret sharing data outsourcing framework. We consider: exact match, range, aggregation and join queries. In each case, we will illustrate how the encryption scheme solves the problem and contrast it with our approach.

Exact Match Queries. An example of an Exact Match Query would be "retrieve the names of all employees whose salary is 20". In the data encryption model [2], [1], all the attributes are encrypted and the service provider only knows the encrypted values. Queries are formulated in terms of the encrypted values, and tuples corresponding to those encrypted values are retrieved. In the secret sharing model, data source D needs to retrieve shares from the service providers. Therefore, it rewrites k queries one for each service provider. For example, the rewritten query for DAS_i would be: Retrieve the tuples of all employees whose salary is $share(20, i)$, where $share(20, i)$ is the share of service provider DAS_i for the secret value 20. In order to find $share(20, i)$, data source D first constructs[1] the polynomial for secret item 20, $p_{20}(x)$, and then it computes the shares, $share(20, i) = p_{20}(x_i)$. After retrieving the corresponding tuples from the service providers, data source D computes the secret values.

Range Queries. In order to process range queries in the encryption model [2], [1], labels are associated with the encrypted tuples. These labels indicate the range of the particular encrypted values. The labels are used to find a superset of the answer in the data encryption method. In order to answer the same query in the secret sharing method, data source D rewrites n queries (one for each of the service provider). For example, the query sent to service provider DAS_i is: All employees whose salaries are between $share(20, i)$ and $share(50, i)$. In order to compute shares, $share(20, i)$ and $share(50, i)$, two order preserving polynomials, $p_{20}(x)$ and $p_{50}(x)$, are constructed ($share(20, i) = p_{20}(x_i)$ and $share(50, i) = p_{50}(x_i)$). Service provider DAS_i, then, sends the tuples of all employees whose salaries are between $share(20, i)$ and $share(50, i)$. Since we have an order preserving polynomial construction technique for the domain, DAS_i can send only the required tuples. After getting this information from the service providers, data source D executes the query by solving the polynomials. Therefore, the computation and communication is performed for those tuples which are required to answer the query. However, the secret sharing scheme does need to communicate with multiple service providers. The consequence of this overhead does result in greater fault-tolerance and data availability in the presence of failures. Future work entails a detailed performance evaluation to determine the computation versus communication trade-off under the two models.

[1]Note that the polynomials are not stored at the data source which would amount to storing the entire data itself. Instead, the polynomials are generated as part of the front-end query-processing at the data source.

Aggregation Queries. We consider *Sum/Average*, *Min/Max/Median* aggregation queries. We classify aggregation queries in two class: 1) Aggregations over Exact Matches 2) Aggregation over ranges. We illustrate aggregation query processing techniques with the following example queries:

- Sum/Average of the salaries of the employees whose name is "John" (Sum/Average over Exact Match).
- Sum/Average of the salaries of the employees whose salary is between 20 and 40 (Sum/Average over Ranges).
- Min/Max/Median of the salaries of the employees whose name is "John" (Min/Max/Median over Exact Match).
- Min/Max/Median of the all salaries of the employees whose salary is between 20 and 40 (Min/Max/Median over Ranges).

Query execution consists of two steps. In the first step, service providers receive the rewritten queries from the data source and perform an intermediate computation. In the second step, the data source receives the intermediate results from all of the service providers and computes the final answer. The above queries are rewritten as follows and sent to the service provider DAS_i :

- Sum/Average of the salaries of the employees whose name is $share('John', i)$.
- Sum/Average of the salaries of the employees whose salary is between $share(20, i)$ and $share(40, i)$.
- Min/Max/Median of the salaries of the employees whose name is $share('John', i)$.
- Min/Max/Median of the all salaries of the employees whose salary is between $share(20, i)$ and $share(40, i)$.

Then, DAS_i finds the tuples needed to answer these queries and performs an intermediate computation over them. After getting all of these intermediate results, data source D computes the final answer.

Join Operations. So far in our development, we assumed data sources have only one table for the sake of the presentation and thus did not consider join operations involving multiple tables. In order to provide database management as a service it is necessary to support join queries over multiple tables. We now demonstrate that the proposed technique can be applied if these tables are related to each other through referential keys and join is based on these keys. Consider a simple schema consisting of two tables *Employees(EID,Name,Lastname,Department,Salary)* and *Managers(EID, ManegerID, ManagerUserName, Password)*. A possible query may ask for the salaries of all managers. To execute this query, these two tables should be joined using the attribute EID. Our scheme can be directly applied to execute this query since join is based on two attributes which are from the same domain and our polynomials are constructed for each domain not for each attribute. Therefore, this join can be done by the service provider at the service provider site. However, if a join is based on two attributes from different domains such as Name and ManagerUserName, then the proposed approach cannot be used for this kind of joins. Thus, the query asking for the salaries of the managers whose name is the same as the ManagerUserName cannot be answered with the proposed scheme. Such extensions and generalizations are the subject of future research and development challenges.

B. Different Types of Data

We have considered only numeric attributes so far and the proposed technique is for numeric attributes. We now briefly explore more complex data types. Of particular interest will be how to represent non-numeric data, and potentially compressed data. We illustrate a simple approach for demonstrating how to apply our scheme for non-numeric attributes. In particular, we need to convert them to numeric attributes. For example, the attribute name length of 5 characters(i.e., VARCHAR(5)), can be represented as a numeric attribute although it is in fact a non-numeric attribute. For the sake of this discussion, assume the characters in names can be one of the letters in the English alphabet and they can be shorter than 5 characters. Thus, the regular expression for this attribute is $(A|B|....|Z|*)^5$ where $*$ represents blank. For instance, name "ABC" is rewritten as "ABC**" while name "FATIH" is rewritten as "FATIH" (because it already has 5 characters). The name attribute consists of a combination of 27 possible characters which are enumerated ($* = 0, A = 1, B = 2, C = 3..., Z = 26$). and thus, each name can represent a number in a number system of base 27. For example, name "ABC**" can be rewritten as $(12300)_{27}$ which corresponds to 21998878 in decimals. With this simple enumeration technique, non-numeric attributes can be converted into numeric attributes and then the proposed outsourcing technique can directly be applied. With the proposed enumeration technique execution of widely used queries over non-numeric attributes can be handled easily. For example, a query asking for employees whose name starts with "AB" or a query asking employees whose name is between "Albert" and "Jack" can be converted into range queries and executed with the range query processing technique discussed earlier.

C. Database Updates

Although our discussion so far has focused on database queries, we would like to note that the proposed techniques are also applicable for database updates. An update would involve retrieving the shares corresponding to the tuples that need to be updated by using the querying approaches discussed above. The actual values for the tuples are reconstructed at the client, and the relevant updates are performed. A new polynomial is constructed and the new shares are distributed to all the service providers. Alternatively, lazy update approaches could be incorporated, and lazy updates as well as incremental updating of values that might reduce the communication overhead are some of the possible directions for future work.

D. Management of Private and Public Data

Once data has been stored at a database service provider, a user may not only want to query his/her own private data, but possibly some public data provided by the database service provider. In fact, the database service provider can thus provide a value added incentive: not only storing the client's private data, but also providing seamless access and integration with a large repository of public data. For example, a client's data may contain her private collection of friends, including information such as phone numbers, addresses, etc. The server may have a database of restaurants and their addresses. The client can exploit the public data to request restaurants that are close to a friend's house, without revealing any private information about the friend, i.e., their name, address, phone number, etc. The combining or *mash-up* of public and private data is especially pertinent in the context of applications arising from national security. Consider the case of an agency such as the FBI that tracks suspicious individuals. Now consider many other public/private agencies such as the TSA that need to correlate the travelers at San Francisco International Airport with the FBI List. This example illustrates the need for *secure mash-up* of public and private data. This problem has recently been addressed in the more limited context of private data stored in a smart USB key [25]. Exploration of the problem of executing database queries on both private and public data in the context of data outsourcing at database service providers remains a formidable challenge.

VI. Concluding Remarks

In this paper, we present scalable secure and privacy-preserving algorithms for data outsourcing. Instead of encryption, which is computationally expensive, we use distribution on multiple data provider sites and information theoretically proven secret-sharing algorithms as the basis for privacy preserving outsourcing. The research is timely due to the ever increasing private and public data being generated. Also, the easy accessibility of data providers on the web makes this paradigm attractive and scalable. Finally, the scientific challenges we have identified are (a) Efficient algorithms for the secure and private execution of different types of database operations, e.g., intersection, join, aggregation, etc. (b) Exploration of different failure models and the development of algorithms for both benign and malicious environments. (c) Algorithms for the management and querying of both private and public data.

Acknowledgment

This research was partially supported by the UC Discovery Grant com-dig05-10189.

References

[1] H. Hacigumus, B. R. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database service provider model," in *Proc. of the ACM SIGMOD Conf.*, 2002.
[2] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," in *Proc. of the VLDB Conf.*, 2004, pp. 720–731.
[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proc. of the ACM SIGMOD Conf.*, 2004, pp. 563–574.
[4] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu, "Two can keep a secret: A distributed architecture for secure database services." in *CIDR*, 2005, pp. 186–199.
[5] M. Kantarcioglu and C. Clifton, "Security issues in querying encrypted data," in *Proc. of the IFIP Conference on Database and Applications Security*, 2005.
[6] B. Hore, S. Mehrotra, and G. Tsudik, "A privacy-preserving index for range queries," in *Proc. of the International Conference on Very Large Data Bases*, 2004.
[7] J. Li and R. Omiecinski, "Efficiency and security trade-off in supporting range queries on encrypted databases," in *Proc. of the IFIP Conference on Database and Applications Security*, 2005.
[8] E. Shmueli, R. Waisenberg, Y. Elovici, and E. Gudes, "Designing secure indexes for encrypted databases," in *Proc. of the IFIP Conference on Database and Applications Security*, 2005.
[9] Z. Yang, S. Zhong, and R. Wright, "Privacy-preserving queries on encrypted data," in *Proc. of the 11 European Symposium on Research In Computer Security*, 2006.
[10] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: Single database, computationally-private information retrieval," in *Proc. of the FOCS*, 1997.
[11] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private infomation retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–982, 1998.
[12] J. Stern, "A new and efiicient all-or-nothing disclosure of secrets protocol," in *Proc. of Asia Crypt*, 1998.
[13] E. Kushilevitz and R. Ostrovsky, "One-way trapdoor permuttions are sufficient for non-trivial single-server private information retrieval," in *Proc. of the EUROCRYPT*, 2000.
[14] C. Cachin, S. Micali, and M. Stadler, "Computationally private information retrieval with polylogarithmic communication," in *Proc. of the EUROCRYPT*, 1999.
[15] Y. Chang, "Single database private information retrieval with logarithmic communication," 2004.
[16] R. Sion and B. Carbunar, "On the computational practicality of private information retrieval," in *Proc. of the Networks and Distributed Systems Security*, 2007.
[17] P. Devambu, M. Gertz, C. Martel, and S. Stubblebine, "Authentic third-party data publication," in *Proc. of the IFIP Workshop on Database Security*, 2000.
[18] E. Mykletun, M. Narasimha, and G. Tsudik, "Authentiction and integrity in outsourced databases," in *Proc. of the ISOC Symposium on Network and Distributed Systems Security*, 2004.
[19] R. Sion, "Query execution assurance for outsourced database," in *Proc. of VLDB Conf.*, 2005.
[20] H. Pang, A. Jain, K. Ramamritham, and K. Tan, "Verifying completeness of relational query resultts in data publishing," in *Proc. of the ACM SIGMOD Conf*, 2005.
[21] M. Narasimha and G. Tsudik, "Authentication of outsourced database using signature aggregation and chaining," in *Proc. of DASFAA*, 2006.
[22] R. Sion, "Secure data outsourcing," in *Proc. of the VLDB Conf.*, 2007, pp. 1431–1432.
[23] T. Ge and S. B. Zdonik, "Answering aggregation queries in a secure system model," in *Proc. of the VLDB Conf.*, 2007, pp. 519–530.
[24] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proc. of the VLDB Conf.*, 2007, pp. 123–134.
[25] N. Anciaux, M. Benzine, L. Bouganim, P. Pucheral, and D. Shasha, "Ghostdb: querying visible and hidden data without leaks," in *Proc. of the ACM SIGMOD Conf.*, 2007, pp. 677–688.
[26] R. Agrawal, A. Evfimievski, and R. Srikant, "Information sharing across private databases," in *Proc. of the ACM SIGMOD Conf.*, 2003, pp. 86–97.
[27] R. Ostrovsky and V. Shoup, "Private Information Storage," in *Proc. of the STOC*, 1997.
[28] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, "Protecting data privacy in private information retrieval schemes," in *Proc. of the STOC*, 1998, pp. 151–160.
[29] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation," in *Proc. of the STOC*, 1999, pp. 245–254.
[30] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[31] F. Emekci, D. Agrawal, and A. E. Abbadi, "Abacus: A distributed middleware for privacy preserving data sharing across private data warehouses," in *ACM/IFIP/USENIX 6th International Middleware Conference*, 2005.
[32] F. Emekçi, D. Agrawal, A. El Abbadi, and A. Gulbeden, "Privacy preserving query processing using third parties." in *ICDE*, 2006, p. 27.