Secure Data Outsourcing Based on Threshold Secret Sharing; Towards a More Practical Solution

Mohammad Ali Hadavi
mhadavi@ce.sharif.edu

Rasool Jalili
jalili@sharif.ir

Network Security Center
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran, +982166166665

ABSTRACT

Database outsourcing is a noteworthy solution to improve quality of services while reducing data management costs. When data is stored and processed out of the territory of its owner, security becomes the first concern. Confidentiality of the outsourced data, correctness assurance of query results, and preserving users' access privacy are the primary requirements of secure data outsourcing. Nevertheless, most of research activities concentrate on confidentiality based on different encryption schemes. This paper reports some aspects of our ongoing research on secure data outsourcing plus our future directions. We propose a framework to provide confidentiality and privacy based on the threshold secret sharing. We discuss the extension points of the framework to satisfy other requirements of secure data outsourcing as well.

1. INTRODUCTION

Storage and retrieving of data has become a big challenge for companies and organizations. Surveys estimate data management cost around five to ten times than that of data gathering [1]. Many organizations prefer to concentrate their human and technical resources on their core business functions rather than data management activities. One nearly new idea is outsourcing data management tasks to an external party. "Database As a Service (DAS)" is an explanation of this approach. Database outsourcing is a noteworthy solution due to reducing data management costs as well as more quality services like database availability. However, it faces with some security challenges.

Data stealing along with commercial competitions challenge the data outsourcing scenario regarding confidentiality aspects. In most cases, the external server hosting the outsourced data and processing queries is untrusted in terms of the data confidentiality. Encrypting outsourced data is the incipient solution to this challenge. However, it imposes extra overhead on the system and decreases the database efficiency. This is the reason that encryption is not considered as a good solution even in centralized in-house databases. Having some noticeable advantages, outsourcing reintroduces encryption-related methods and techniques to satisfy its requirements. To overcome the efficiency problem when using encryption-based methods, researchers investigate mechanisms for direct execution of queries on the encrypted data. Such mechanisms provide confidentiality in addition to efficiency as they do not typically decrypt data at the server side in order to execute a query.

Query result correctness is another challenge in secure data outsourcing when the external server is not honest. This issue has not been investigated as much as confidentiality issue in database outsourcing.

Beside encryption-based approaches, secret sharing as another approach indicates a promising prospect to solve the challenges of secure data outsourcing. This paper studies application of secret sharing in secure data outsourcing. Hence, a secret sharing based framework providing the confidentiality of data has been introduced. Then, future work for validation, extension, or improvement of the framework has been mentioned.

The remainder of this paper is organized as follows: Section 2 discusses basic requirements of secure data outsourcing. Section 3 briefly reviews related work. Section 4 proposes a preliminary framework based on the threshold secret sharing scheme. Section 5 discusses areas of future work to complete the proposed framework in order to satisfy other considerations of the DAS model. Finally, Section 6 summarizes and concludes the paper. For simplicity, we may refer to Secure Data Outsourcing as SDO in the remainder of this paper.

2. BASIC REQUIREMENTS OF A SDO SOLUTION

Despite almost a decade of research on SDO, the proposed approaches have not been reported to have much success in operation in the actual community/industry. This has been basically due to many restrictions and overheads accompanied with the current approaches. In other words, the intrinsic requirement for SDO is to improve the existing models regarding practicality aspects while considering security considerations. To elaborate, we enumerate these requirements with respect to security and practicality aspects in the following bullets. The first four items focus on security concerns and the last three ones concentrate on practicality issues.

- Identification of the level of trust in the server: The majority of existing approaches assume that the server is untrusted in terms of data confidentiality but trusted in terms of query result correctness. A more general solution should downgrade

VLDB 2010 PhD Workshop, September 13, 2010, Singapore

the trust level and assume the server is untrusted in both confidentiality and correctness aspects. So, the existing models should be extended or new models be proposed to assure the query result correctness.

- Protection against security attacks: Security threats and attacks are one of the most important aspects which should be considered while proposing models and frameworks for SDO. Basically, the more tolerable the model is, the more secure solution could be achieved. Known database attacks, frequency attacks, size based attacks, and collusion are among the important threats in database outsourcing scenario.
- Users' access privacy: Users' access privacy concerns the relation between users and their queries. Privacy is one of the primary requirements for SDO. Intruders as well as untrusted server cannot infer wealthy information from the user's queries while privacy is preserved.
- Access control policy enforcement: In operational environments, users have different access rights to database entities. Enforcement of access control policies is one of the important aspects of SDO. For efficiency reasons, the policies should be enforced by the server. On the other hand, for security reasons, the policies may be confidential and untrusted server should not be aware of or infer about them.
- Supporting different kinds of queries: Proposed models for SDO will be practicable if it supports execution of different kinds of queries. Exact match queries, pattern matching for string data, range queries, JOIN queries, and queries contain aggregation functions like SUM, AVG, COUNT, MIN, and MAX are among typical queries and should be supported.
- Supporting multiple and variant data types: Existing databases contain a variety of data types, nevertheless most of existing approaches for SDO concentrate on numeric data. String data types, multimedia data types, and data types with a limited number of values (like Boolean and Enumerated types) should also be considered in order to have a realistic solution for data outsourcing.
- Efficiency besides acceptable overhead: We need methods, models, and techniques that are efficient besides having acceptable overhead. In the context of SDO, storage, communication, and computational overhead are important. Computational and storage overhead should be considered at the client and the server sides. Communication overhead refers to the amount of interaction between a client and a server (the number of passes between the client and the server and the amount of information to be transferred) while processing a query. Those methods in which the sever returns extra records to the client as false hit records, introduce more communication overhead and also computational overhead at the client side. In such cases, clients need some computational power to purge the received results based on the user query.

3. RELATED WORK

Most of research activities in the area of secure data outsourcing concentrate on confidentiality of the outsourced data against untrusted servers.

Hacigümüs *et al.* [27] explored the DAS model and then proposed a solution for executing queries over encrypted data concerning confidentiality problem [2]. In their solution, a query is processed in four steps. At first, the user submits a query through a client. Then, the client transforms it into a new query that is executable on encrypted data at the server side. The server executes the query and sends the results for the client. The client purges the results and returns final results to the user. Hacigümüs *et al.* [3] improved their work concerning query optimization. They proposed a privacy homomorphism encryption scheme to support aggregation queries in their model [4]. Mykletun *et al.* [5] showed these encryption functions are susceptible to known plain text attacks and reveal the encryption key. They proposed another solution to support aggregation queries in their work.

The basic idea of [2, 3] is bucketing and indexing of attribute values. There are different ways for bucketing values such as equivalence-width, equivalence-depth, and using hash functions proposed on numeric and character data [6, 7, and 8]. The index production methods should consider query processing efficiency while maintaining security. An adversary as well as an untrusted server should not be able to infer about the plain data using assigned indexes.

However, index based methods, due to their admissible efficiency, are the most popular methods for confidentiality problem in data outsourcing. There are some works on security analysis [10, 11] and reducing false hits [12] for index based methods.

Some research activities to support range queries have been done. Damiani *et al.* [13] used B^+-tree to support range queries. Agrawal *et al.* [14] proposed an Order Preserving Encryption Scheme (OPES) for numeric data to support range queries. The OPES encryption preserves the order of plain data in encrypted data, so it is susceptible to known database attacks. Mansouri [26] added randomness to OPES and proposed the ROPES as Random Order Preserving Encryption Scheme to protect from known database security attacks.

Agrawal *et al.* [1] proposed a new approach to provide confidentiality of outsourced data using the idea of secret sharing. They used Shamir threshold secret sharing scheme [15] to provide confidentiality in database outsourcing. In the next section, we extend their method and propose a preliminary framework. Further, we give some possible future work to show this new approach could satisfy basic requirements of SDO. Brinkman *et al.* [16, 17] used secret sharing to store and query tree structured data (such as XML) securely.

Yong *et al.* [18] concentrate on privacy requirement. Preserving privacy causes an intruder or an untrusted server not to be able to gain valuable information about queries and users who submit the queries.

Correctness requirement of SDO refers to integrity and completeness of query results. Integrity assurance means an untrusted server cannot tamper the query results in an unauthorized manner. Completeness means all of the result set is delivered to the client. Min Xie *et al.* [19] provide audit-ability of correctness via adding a limited number of records to the outsourced database. These extra records can be produced by randomized or deterministic methods. They prove security of their method. Mykletun *et al.* [20] proposed integrity mechanisms for simple SELECT queries. Merkle tree and cryptographic hashing

structures have been used in [21] to provide correctness verification in range queries.

4. OUR PROPOSED FRAMEWORK

We use Shamir's threshold secret sharing scheme [15] to share a secret (data to be outsourced) between *n* servers. We call these servers "*Data Servers*". Subsequently, data indexes are maintained in a separate server called "*Index Server*". So, clients interact with two kinds of servers. Data shares are stored on *Data Servers* and referenced by *Index Server*. To distribute a secret (an attribute value) between *n Data Servers* the distributor (data owner) selects a vector $X=\{x_1, x_2, \ldots, x_n\}$ and a polynomial $f(x)$ of order $k-1$ such that the constant value of $f(x)$ be equal to the secret. Other coefficients are determined randomly. Regarding randomness of $f(x)$'s coefficients for each attribute value, the order of values is not preserved, i.e. a greater value may have lower share than a lesser one, or two equal values may have different shares. This property provides protection from frequency attacks on *Data Servers.*

In the threshold secret sharing with the threshold k of n, we need at least k shares to retrieve the secret. Secret v_s is calculated by the Lagrange interpolation with the following formula:

$$v_s = \sum_{j=1}^{k} \left(f\left(x_{i_j}\right) \prod_{\substack{\mu=1 \\ \mu \neq j}}^{k} \frac{0 - x_{i_\mu}}{x_{i_j} - x_{i_\mu}} \right)$$

4.1 Data Servers

In a *threshold (k, n)* secret sharing, *Data Servers* store data shares computed by $f(x)$ of order $k-1$ and vector X. For each attribute value to be shared between *Data Servers*, $f(x)$ with the order of $k-1$ is created. The constant value of $f(x)$ is equal to the to-be-shared value and the other coefficients are randomly selected. For each value, $Share_i$, denoting the secret share of i^{th} *Data Server* $(1 \leq i \leq n)$, is calculated by $f(x_i)$ where, x_i is the i^{th} member of the vector X. X is owned by the data owner and kept hidden from untrusted servers. Hence, even if more than k *Data Servers* accumulate their shares, they cannot retrieve the secret. In this approach, known database attacks are prevented as the order of shares in *Data Servers* does not follow the real order of attribute values due to randomized coefficients of $f(x)$.

4.2 Index Server

To build indexes at *Index Server*, attribute values are encrypted using an order preserving encryption scheme. Then, a B^+-tree is made over the encrypted values. Each leaf in the B^+-tree refers to a bucket containing record numbers of *Data Servers* with the same share value of the corresponding leaf key. Then, the buckets are encrypted and sent with the B^+-tree to *Index Server*. Thereafter, *Index Server* is responsible for any insertion, deletion, or update in the B^+-tree.

In order to insert a new record into database, the client needs to interact with *Data Servers* and *Index Server*. At first, share values of the record attributes are calculated and inserted into *Data Servers* as a new record. Subsequently, the B^+-tree in *Index Server* is modified for the indexed attributes. In order to modify the B^+-tree for each indexed attribute, the new attribute value is encrypted with the "order preserving encryption scheme" and sent to *Index Server*. *Index Server* locates the appropriate location in the tree and inserts it in that location. If the inserted value is repetitive, there is a leaf with the same key in the tree. In such a case, simply the new record number is appended to the corresponding bucket. If the value is not repetitive, a new leaf is inserted into the B^+-tree and its bucket is created with the record number written into it. In order to append the record number in the corresponding bucket, the bucket is sent to the client. The client decrypts the bucket which the item is inserted into. Then, the bucket is encrypted and sent to *Index Server*.

Record removal is done by removing the related record number from the appropriate bucket in the B^+-tree. So, frequent garbage collection is done for physical deletion of records in *Data Servers*. Therefore, we need a method to distinguish logically deleted records in *Data Servers*.

Although, the order of attribute values is preserved in *Index Server*, adversary (that might be *Index Server* itself) cannot infer any relation between encrypted values on *Index Server* and data shares on *Data Servers* as buckets are encrypted. Notwithstanding, *Index Server* might infer about the attribute values frequency from the buckets volume. We use padding to assimilate bucket volumes in order to prevent this threat.

4.3 Query Processing

The query processing scenario in our approach is as follows. The values in the query predicate are encrypted in the order preserving manner and sent to *Index Server*. *Index Server* searches for these values in the related B^+-tree and sends the corresponding buckets to the client. The client decrypts the buckets and requests from *Data Servers* to retrieve records with the numbers extracted from decrypted buckets. When, at least k *Data Servers* reply to the client, the final result can be retrieved by the client through Lagrange interpolation.

As we saw in this scenario, *Data Servers* cannot be informed about the query contents. They act as storage resources which return records only based on their numbers. There are not any false hits in returned results from *Data Servers*. Therefore, neither the client nor *Data Servers* calculate data shares on *Data Servers*. So, we do not need to store and find coefficients of the polynomial $f(x)$.

Our approach provides for both exact match and range queries. Range queries are performed via search on the order preserved encrypted B^+ tree index. Aggregation queries are also supported due to additive homomorphism property of secret sharing. Queries contained MIN, MAX, and COUNT functions are executed at *Index Server* without *Data Servers* interference.

4.4 Example

Consider the simple Employee relation in Table 1 with four attributes ID, Name, Age, and Salary. For simplicity, assume we restrict ourselves to search merely based on Age attribute. So, a B^+-tree is made on Age values. This B^+-tree is shown in Figure 1. In order to support range queries, the rightmost pointer of each leaf refers to the next leaf, except the last leaf whose rightmost pointer is null. Leaves have pointers to buckets consisting of the appropriate record numbers. E_{op} in Figure 1 denotes order preserving encryption used in B^+-tree structure and E is an arbitrary encryption function used to encrypt buckets.

Table 1. The Employee relation

ID	Name	Age	Salary
1	Elvis	45	100
2	John	84	200
3	Mary	78	150
4	Frank	46	350
5	Bob	45	200
6	Alice	80	210
7	Henry	45	175
8	Jack	57	220
9	Gary	57	300
10	Donna	46	130

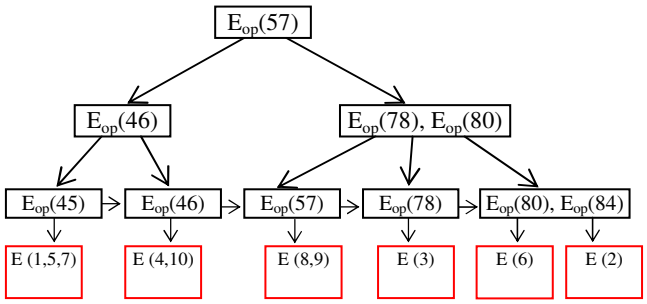

Figure 1. B$^+$-tree index on the Age attribute

Let $n=3$, $k=2$, and $X = \{x_1=3, x_2=1, x_3=5\}$. So, the order of $f(x)$ is $k-1=1$ and the number of *Data Servers* is three. The client will be able to retrieve the results of a query, if it receives response from at least two *Data Servers*.

At *Data Server* side, each *Data Server* stores data shares of each value for all confidential attributes. For simplicity, in this example we only show the Salary values in *Data Servers*. We have ten records in the Employee table, so the data owner creates ten polynomials of the general form ax_i+b; where x_i is selected from the X vector for each *Data Server* and a is obtained randomly for each polynomial. The data owner does not need to store these random numbers. b is equal to the Salary value for each polynomial. Figure 2 depicts the polynomials and the calculated data shares in *Data Servers*. For simplicity, we assume a values are integers between one and ten.

Suppose a user submits the query "SELECT Salary FROM Employee WHERE Age=46". Then, the corresponding client sends $E_{op}(46)$ to *Index Server*. *Index Server* locates the value and returns the corresponding bucket to the client. The client decrypts the bucket and accesses 4 and 10 as record numbers. Then, the client requests these records from *Data Servers*. In our example, Data Server 1 sends 374 and 133, Data Server 2 sends 358 and 131, and Data Server 3 sends 390 and 135 as the forth and tenth records respectively. When at least two *Data Servers* respond to the client, attribute values are retrievable through the Lagrange interpolation. Assume the results are received from *Data Server* 1 and *Data Server* 3. The client puts the values in the Lagrange formula and calculates the Salary for the forth and tenth records. For example, the value 350 is obtained for the forth record by the following relation:

$$v_s = 374\frac{0-5}{3-5} + 390\frac{0-3}{5-3} = 350$$

The client does a similar calculation to retrieve salary value of the tenth record.

Now consider the following query with the SUM aggregation function: "SELECT SUM (Salary) FROM Employee WHERE Age > 50". To execute this query, the client sends $E_{op}(50)$ to *Index Server*. *Index Server* returns those buckets referred by leaves with values greater than 50 as their keys to the client. In this example, four buckets are returned by *Index Server*. The client decrypts them and sends the obtained record numbers to *Data Servers*. Each *Data Server* calculates the sum of its shared values for requested records and sends the result back to the client. Finally, the client retrieves the final result by interpolating received values due to additive homomorphism of secret sharing. Such a property can be examined easily in this example.

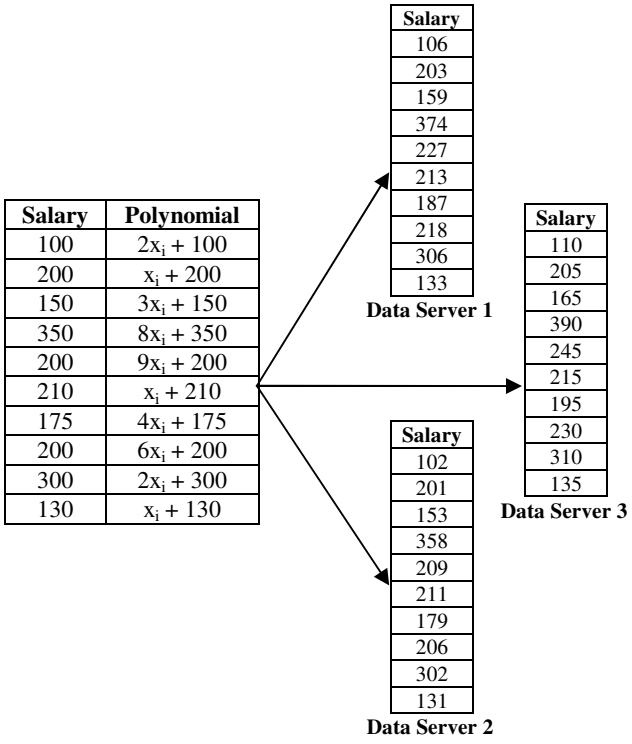

Figure 2. Sharing Salary values among three *Data Servers*

4.5 Comparison

In this section we compare our proposed framework with some other index-based methods [2, 3, 10, 12, and 13], encryption-based methods [4, 5, and 14] and also secret-sharing-based Agrawal *et al.*'s proposed method [1].

Similar to the Agrawal *et al.*'s method [1], our approach supports different kinds of query including exact match queries, range queries, and aggregation queries. Update, delete, and insert operations are also supported. Except the proposed scheme by Hacigümüs *et al.* [4] and the one proposed by Mykletun *et al.* [5], the other published index-based methods do not support aggregation queries. Nevertheless, their ([4] and [5]) support for aggregation queries is either insecure or inefficient. Supporting

range queries in index-based methods depends on the bucketing strategy and the index construction method.

Index based methods usually have some limitations for update, insert, and delete operations. For example, the distribution of values is used for bucketing and index construction. These operations change the distribution and render the rebucketization necessary. This is a time consuming work that makes index based methods suitable for read-only data. Agrawal *et al.*'s method for supporting range queries [14] is not secure against the known database attacks. It has also the mentioned problem for update, delete, and insert operations. This problem requires frequent re-encryption due to adopting new data distribution. As described in section 4, our approach supports the update, delete and insert operations efficiently.

In our approach, similar to the Agrawal *et al.*'s method [1], the received results from *Data Servers* do not have any false hits. This could be considered as an advantage since it reduces the communication and computational overhead at the client side. Other index based methods [2, 3, and 12] produce false hits in the server results. It is worth mentioning that those methods which do not generate false hits at the server side face with security challenges due to the frequency and known database attacks. Evdokimov *et al.* [9] have proven that index based methods especially the methods not generating false hits at the server side are not secure. Damiani *et al.* [13] proposed a secure index based method, but it has a large amount of communication overhead in its query processing scenario. However, while our approach has a considerable level of prominent security, it also enjoys the advantages of not generating false hits and an acceptable communication overhead compared to that of Damiani *et al.*'s research.

In secure data outsourcing, usually there is a tradeoff between efficiency and security. To the best of our knowledge, no secure model for data outsourcing with high efficiency has been reported yet. Our approach, while reducing the computational cost due to usage of distribution and interpolation instead of encryption and decryption functions, provides high level of security. Although computational overhead in our approach is more than Agrawal *et al.*'s scheme [1], it offers much more security. Our approach is secure against known database and frequency attacks. The secret data is retrievable only by the client who knows the X vector. Thus, the collusion problem is addressed, i.e. *Data Servers* cannot get the secret even if they collude with each other or with the *Index Server* to reveal the secret.

5. FUTURE WORK

A preliminary framework based on the idea of threshold secret sharing was proposed in Section 4. Confidentiality and privacy as the basic requirements of SDO have been considered in our framework. Regarding efficiency and supporting different kinds of queries, we think this framework is a starting point for a more realistic solution for secure data outsourcing. We believe that extending and improving this framework to satisfy the basic requirements of this area can help the DAS model to be more practical. In this section, we review the possible extension points of our approach as future work.

- Reducing the trust level to external servers: While major proposed methods suppose the server is honest but curious, we try to decrease the trust level and assume the server is untrusted in terms of both confidentiality and correctness. So, in addition to providing confidentiality, integrity and completeness of results must be also assured. The threshold secret sharing scheme, provides the potentiality of correctness validation and distinguishing malicious servers in regard to the distribution of secrets among n data servers and retrieving form k ones $(k \leq n)$. A model for correctness validation with assumptions like potentiality of collusion between untrusted servers is an extension point of the proposed framework to develop a plenary solution for SDO.

- Formal proof for security: The proposed solution must tolerate different threats and security attacks. Formal proof for security in terms of confidentiality, privacy, and correctness is required to complete the proposed approach.

- Supporting different data types: In this study we concentrated on numeric data in our framework. Considering different data types especially character data has a pivotal role in practicality of SDO. Data types such as Boolean that accept limited number of values and multimedia which are rational to be confidential in some modern applications should also be explored. Fortunately, noticeable research conducted on the area of multimedia secret sharing promises acceptability of these data types in our framework.

- Database access control policy enforcement: We can extend our framework to adopt the ability of applying access control policies in multi user environments based on the secret sharing concept. Fortunately, the secret sharing has been shown as one of the basic concepts for structuring access control mechanisms [22, 23, and 24]. Enforcement of policies by untrusted server (probably with the slight data owner involvement), update-ability of the policies, and preserving privacy of the access control policy for untrusted server are the most challenging issues to adopt an access control policy enforcement mechanism in database outsourcing scenario.

  Considering the above challenges, we have done some research in the area of access control for SDO scenario which cannot be included in this paper due to restrictions on number of pages.

- Availability, efficiency, and security tradeoff: in the (k, n) threshold secret sharing, data is distributed among n servers and retrieved from at least k ones. The values of k and n affect different aspects such as availability and efficiency, and also the storage, communication, and computational overheads. Study about these aspects to find a tradeoff point based on the user requirements is another future work to produce a desirable model.

6. CONCLUSION

Secure data outsourcing is in its adolescence and notwithstanding reported research activities, is far away from being practical regarding its goals and requirements. Secure data outsourcing refers to preserving confidentiality of outsourced data, privacy of users' queries, and query result correctness assurance [25].

We believe that the idea of secret sharing with respect to strong theoretical security background on the one hand, and less computational overhead rather than encryption on the other hand,

indicates promising future for secure data outsourcing. Our secret sharing based framework provides a basis to support different kinds of queries on different data types. Additive homomorphism property of secret sharing supports efficient execution of a wide range of aggregation queries. Although communication and storage overhead in secret sharing schemes seems to be high, availability and correctness verifiability are potentially obtained. Storage overhead in this approach is almost comparable to database servers that use replication for availability and disaster recovery reasons.

It seems the idea of secret sharing for secure data outsourcing, can satisfy the basic requirements mentioned in section 2. In this regard, concentration on adopting secret sharing for string data without unrealistic simplification and constraints, tradeoff analysis and security proofs based on parameters such as the number of servers and availability measures, the correctness assured model for the outsourced data, and access control policy enforcement in an outsourced database are among future work in this area.

7. REFERENCES

[1] D. Agrawal, A. El Abbadi, F. Emekci, A. Metwally, Database management as a service: challenges and opportunities. In *Proceedings of IEEE International Conference on Data Engineering (ICDE '09).* IEEE Computer Society, 2009, 1709-1716.

[2] H. Hacigümüs, B. Lyer, C. Li, S. Mehrota, Executing SQL over encrypted data in the database-service-provider model. In *proceedings of the ACM SIGMOD International Conference on Management of Data* (Madison, Wisconsin). ACM, 2002, 216-227.

[3] H. Hacigumus, B. Iyer, S. Mehrotra, Query optimization in encrypted database systems. In *Proceedings of 10th International Conference on Database Systems for Advanced Applications (DASFAA' 05)*. Springer, 2005, 43-55.

[4] H. Hacigumus, B. Iyer, S. Mehrotra, Efficient execution of aggregation queries over encrypted relational databases. In *Proceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA' 04)*. Springer, 2004, 2973, 125–136.

[5] E. Mykletun, G. Tsudik, Aggregation queries in the Database-As-a-Service model. In *Proceedings of 20th Annual IFIP WG 11.3 working conference on data and applications security (DBSEC' 06)*, Springer, 4127, 2006, 89-103.

[6] Z. Wang, J. Dai, W. E. I. Wang, B. Shi, Fast query over encrypted character data in database. *Communications in Information and Systems*, 2004, 4, 289-300.

[7] Y. Zhang, W. Li, X. Niu, A method of bucket index over encrypted character data in database. In *Proceedings of Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP' 07),* IEEE Computer Society, 2007, 186-189.

[8] E. Goh, *Secure Indexes.* Cryptography ePrint Archive, Report 2003/216, 2003.

[9] S. Evdokimov, M. Fischmann, O. Gunther. Provable security for outsourcing database operations. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE '06)*. IEEE Computer Society, 2006, 117.

[10] B. Hore, S. Mehrotra, G. Tsudik. A privacy-preserving index for range queries. In *Proceedings of the 30th VLDB Conference*. VLDB Endowment, 2004, 720-731.

[11] A. Ceselli, E. Damiani, S. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Modeling and assessing inference exposure in encrypted databases. *ACM Trans. Information and System Security (TISSEC), 8*, 2005, 119-152.

[12] Y. Tang, J. Yun. A method for reducing false hits in querying encrypted databases. In *Proceedings of the 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC-EEE' 06).* 2006, 22.

[13] E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In *Proceedings of the 10th ACM Conference on Computer and Communications Security.* ACM, 2003, 93-102.

[14] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data.* ACM, 2004, 563-574.

[15] A. Shamir. How to share a secret. In *Communications of the ACM*, 1979, 22, 11, 612–613.

[16] R. Brinkman, J. M. Doumen, P. H. Hartel, W. Jonker. Using secret sharing for searching in encrypted data. In *Workshop on Secure Data Management in a Connected World (SDM' 04)*. Springer, LNCS 3178, 2004,18–27.

[17] R. Brinkman, B. Schoenmakers, J. M. Doumen, W. Jonker. Experiments with queries over encrypted data using secret sharing. In *Secure Data Management VLDB 2005 workshop*. Springer, LNCS 3674, 2005, 33-46.

[18] Z. Yang, S. Zhong, R.N. Wright. Privacy-preserving queries on encrypted data. In *Proceedings of the 11th European Symposium on Research in Computer Security (Esorics' 06)*. Springer, LNCS 4189, 2006, 479-495.

[19] M. Xie, H. Wang, J. Yin, X. Meng. Integrity auditing of outsourced data. In *proceedings of the* 33rd international conference on Very large data bases *(VLDB '07)*. VLDB Endowment, 2007, 782-793.

[20] E. Mykletun, M. Narasimha, G. Tsudik. Authentication and integrity in outsourced databases. *ACM Trans. Storage (TOS)*, 2, 2 (May 2006), 107-138

[21] H. Pang, A. Jain, K. Ramamritham, K. Tan. Verifying completeness of relational query results in data publishing. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (Baltimore, Maryland). ACM, 2005, 407-418.

[22] S. Liu, W. Li, L. Wang. Towards efficient over-encryption in outsourced databases using secret sharing, In *Proceedings of The 2nd IFIP International Conference on New Technologies, Mobility and Security (NTMS 2008)*. IEEE Press, 2008, 1-5.

[23] C. Lin, W. Lee. Efficient secret sharing with access structures in a hierarchy. In *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)* (Volume 2), IEEE Computer Society, 2005, 123-126.

[24] J Marti-Farré, C. Padro. Secret sharing schemes on access structures with intersection number equal to one. *Journal of Discrete Applied Mathematics*, 153, 3 (March 2006), 552-563.

[25] R. Sion. Secure data outsourcing. In *Proceedings of the 33rd international conference on Very large data bases (VLDB' 07)*. VLDB Endowment, 2007, 1431–1432.

[26] F. Mansouri. *A method for executing queries on encrypted databases without firstly decrypting them.* MSc thesis, Sharif University of Technology, December 2008 (in Persian).

[27] H. Hacigümüs, B. Iyer, S. Mehrotra. Providing database as a service, In *Proceedings of 18th International Conference on Data Engineering (ICDE' 02).* IEEE Computer Society, 2002, 29.