Ensuring Correctness, Completeness, and Freshness for Outsourced Tree-Indexed Data

Tran Khanh Dang, National University of Ho Chi Minh City, Vietnam

ABSTRACT

In outsourced database service model, query assurance takes an important role among well-known security issues. To the best of our knowledge, however, none of the existing research work has dealt with ensuring the query assurance for outsourced tree-indexed data. To address this issue, the system must prove authenticity and data integrity, completeness, and freshness guarantees for the result set. These objectives imply that data in the result set is originated from the actual data owner and has not been tampered with; the server did not omit any tuples matching the query conditions; and the result set was generated with respect to the most recent snapshot of the database. In this paper, we propose a vanguard solution to provide query assurance for outsourced tree-indexed data on untrusted servers with high query assurance and at reasonable costs. Experimental results with real datasets confirm the efficiency of our approach and theoretical analyses.

Keywords: *database services outsourcing; dynamic search trees; query assurance; security and privacy; untrusted servers*

INTRODUCTION

Outsourcing database services is emerging as an important new trend thanks to continued growth of the Internet and advances in the networking technology. Organizations outsource their data management needs to an external service provider, thereby freeing them to concentrate on their core business. In this outsourced database service (ODBS) model, organizations rely on the premises of external service providers, which include hardware, software, and manpower, for the storage and retrieval management of their data, and they operate other business applications via the Internet without having to maintain applications *in-house.* Figure 1 depicts key "actors" in the most general and complicated ODBS model (Mykletun, Narasimha, & Tsudik, 2004),[1] where multiple data owners (say, separate departments of an organization) outsource their data to a certain database server (which may be untrusted) and allow users (may be other departments, partners of the organization, or even themselves) to access the outsourced data. This service model is a recent and important manifestation of the outsourcing trend of different information

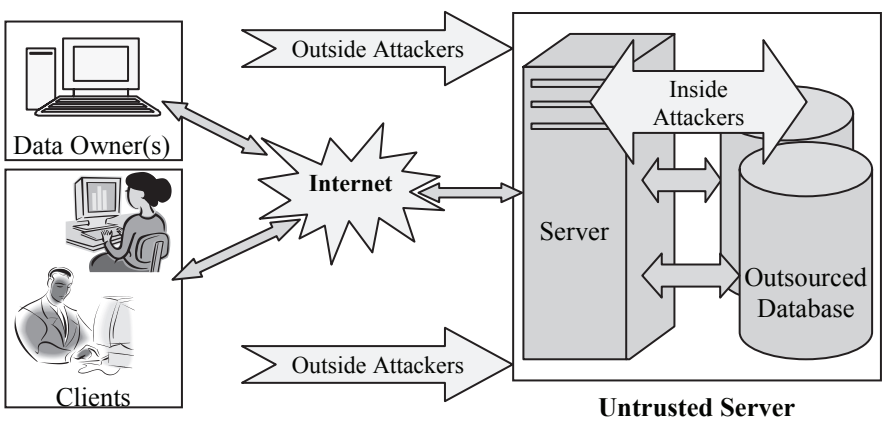
technology services. As we can see, however, among issues needing to be addressed in order to make this model reality, security-related issues must be of crucial concern due to the fact that the server may be untrusted, and both data as well as users' queries can now be exposed to the server and hackers/malicious users (corresponding to inside and outside attackers as shown in Figure 1, respectively). This means that, in this ODBS model, apart from secure network communication channels and other necessary security procedures at the user side (Axelrod, 2004), efficient and effective solutions to security threats inside the server are indispensable. We discuss in more detail these server-side security-related issues below.

Security Issues in the ODBS Model

Since a service provider is typically not fully trusted, the ODBS model raises numerous interesting research challenges related to security issues. First of all, because the life-blood of every organization is the information stored in its databases, making outsourced data confidential is therefore one of the foremost challenges in this model. In addition, privacy-related concerns must also be taken into account due to their important role in real-world applications.[2] Not less importantly, in order to make the outsourced database service viable and really applicable, the query result must also be proven qualified. This means the system has to provide users with some means to verify the query assurance claims of the service provider. Overall, most crucial security-related research questions in the ODBS model relate to the below issues:

- **Data confidentiality:** Outsiders and the server's operators (database administrator–DBA) cannot see the user's outsourced data contents in any cases (even as the user's queries are performed on the server).
- **User privacy:** Users do not want the server and even the DBA to know about their queries and the results. Ensuring the user privacy is one of the keys to the ODBS model's success.
- **Data privacy:** Users are not allowed to get more information than what they are querying on the server. In many situations, users must pay for what they have got from the server and the data owner does not allow them to get more than what they have paid for, or even users do not want to pay for what they do not need because of the low bandwidth connections, limited memory/storage devices, and so forth. This security objective is not easy to obtain and a cost-efficient solution to this issue is still an open question (Dang, 2006b).
- **Query assurance:** Users are able to verify the correctness (authenticity and data in-

Figure 1. The ODBS model and security threats at the server side



tegrity), completeness, and freshness of the result set. Among all security objectives, the query assurance is *always* appealed in the ODBS model. We succinctly explain these concepts as follows, and more discussions can be found in Narasimha and Tsudik (2006), Mykletun et al. (2004), Boneh, Gentry, Lynn, and Shacham (2003), Pang and Tan (2004), Pang, Jain, Ramamritham, and Tan (2005), and Sion (2005):

- **Proof of correctness:** As a user queries outsourced data, it expects a set of tuples satisfying all query conditions and also needs assurance that data returned from the server originated from the data owner and have not been tampered with either by an outside attacker or by the server itself.
- **Proof of completeness:** As a user queries outsourced data, completeness implies that the user can verify that the server returned *all* tuples matching all query conditions, that is, the server did not omit any tuples satisfying the query conditions. Note that a server, which is either malicious or lazy, might not execute the query over the entire database and might return no or only partial results. Ensuring the query result completeness aims to detect this unexpected behavior.
- **Proof of freshness:** The user must be ensured that the result set was generated with respect to the most recent snapshot of the database. This issue must be addressed so as to facilitate *dynamic* outsourced databases, which frequently have updates on their data.

The above security requirements differ from the traditional database security issues (Castano, Fugini, Martella, & Samarati, 1995; Umar, 2004) and will in general influence the performance, usability, and scalability of the ODBS model. Although there exist a number of research works on the above topics (e.g., Du & Atallah, 2000; Hacigümüs, Iyer, Li, & Mehrotra, 2002a; Bouganim & Pucheral, 2002; Damiani, Vimercati, Jajodia, Paraboschi, & Samarati, 2003; Lin & Candan, 2004; Chang & Mitzenmacher, 2004; Dang, 2006a, 2006b), to the best of our knowledge, none of them has dealt with the problem of ensuring *query assurance* for *outsourced tree-indexed data.* It has been clearly proven in the literature that tree-indexed data have played an important role in both traditional and modern database applications (Dang, 2003). Therefore, security issues in query assurance for outsourced tree-indexed data need to be addressed completely in order to materialize the ODBS model. This is even then not a trivial task, especially as tree-based index structures are outsourced to untrusted servers (Du & Atallah, 2000; Dang, 2005). In this article, we will discuss and propose solutions to security issues in order to provide query assurance for *dynamic* outsourced databases that come together with tree-based index structures. Our techniques allow users to operate on their outsourced tree-indexed data on untrusted servers with high query assurance and at reasonable costs. Our proposed solutions will address all three desired security properties of query assurance.

In addition, as presented in Du and Atallah (2000), Mykletun et al. (2004), and Dang (2006b), there are a number of different ODBS models depending on desired security objectives. In this article, however, due to the complexity of the big problem, we will focus on the most basic and typical ODBS model where only *data confidentiality, user privacy,* and *query assurance* objectives should be taken into account. Our holistic solution allows users (also the data owners in our considered ODBS model) to manipulate their outsourced data as it is being stored in in-house database servers.

The rest of this article is organized as follows. We briefly summarize main related work and introduce a state-of-the-art approach to managing outsourced tree-indexed data without query assurance considerations. After that, we present our contributions to completely solve the problem of query assurance in dynamic outsourced tree-indexed data. We then show

experimental results with real datasets and brief security analyses in order to establish the practical value of our proposed solutions. Finally, the last section gives conclusions and future work.

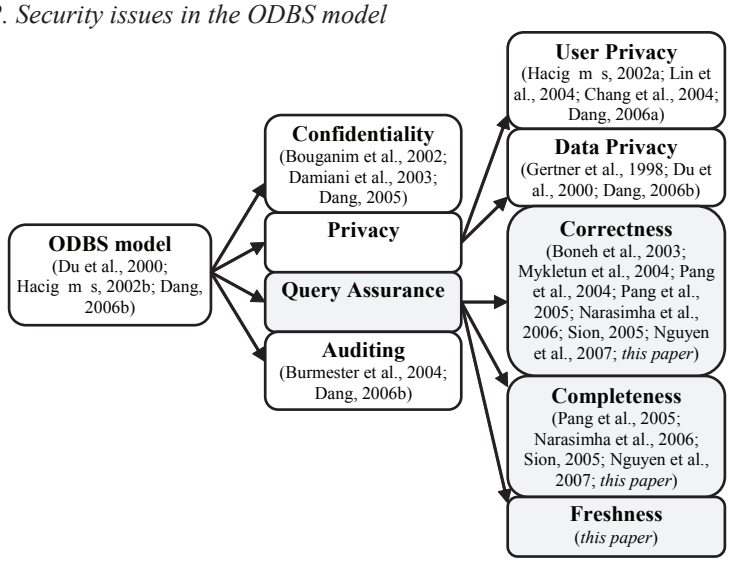
RELATED WORK

Although various theoretical problems concerning computation with encrypted data and searching on encrypted data have appeared in the literature (Fong, 2003), the ODBS model which heavily depends on data encryption methods emerged not long ago (Du & Atallah, 2000; Hacigümüs, Mehrotra, & Iyer, 2002b; Dang, 2006b). Even then it rapidly got special attention from the research community due to a variety of conveniences brought in as well as interesting research challenges related (Dang, 2005). The foremost research challenge relating to security objectives for the model was introduced in the previous section. In Figure 2 we diagrammatically summarize security issues in the ODBS model, together with major references to the corresponding state-of-the-art solutions.

As shown in Figure 2, most security objectives of the ODBS model have been investigated. To deal with data confidentiality issue, most approaches adopted to encrypt (outsourced) data before its being stored at the external server (Bouganim & Pucheral, 2002; Damiani et al., 2003; Dang, 2005). Although this solution can protect the data from outsiders as well as the server, it introduces difficulties in the querying process, as it is hard to ensure the user and data privacy when performing queries over encrypted data.

In general, to address the privacy issue (including both user and data privacy), outsourced data structures (tree or non-tree based) that are employed to manage the data storage and retrieval should be considered. Notably, the problem of *user privacy* has been well solved, even without special hardware (Smith & Safford, 2001), if the outsourced database contains only encrypted records and no tree-based indexes are used for storage and retrieval purposes (see Dang, 2006b, for an overview). Conversely, the research result is less incentive if such trees are employed, although some proposals have been made recently (Lin & Candan, 2004; Dang, 2006b). In our previous work (Dang, 2006b), we did propose an *extreme protocol* for the ODBS model based on private information retrieval (PIR)-like protocols (Asonov, 2001). It would, however, become prohibitively expen-

Figure 2. Security issues in the ODBS model



sive if only one server were used to host the outsourced data (Chor, Goldreich, Kushilevitz, & Sudan, 1995). Damiani et al. (2003) also gave a solution to query-outsourced data indexed by B^+-trees, but their approach does not provide an obvious way to traverse the tree, and this may lead to compromised security objectives (Lin & Candan, 2004; Dang, 2006a). Lin and Candan (2004) introduced a computational complexity approach to solve the problem with sound experimental results reported. Their solution, however, only supports obvious search operations on outsourced search trees, insert, delete, and modification ones. That means their solution cannot be applied to *dynamic* outsourced search trees where several items may be inserted into and removed from, or existing data can be modified. In our recent work (Dang, 2006a), we analyzed and introduced techniques to completely solve the problem of data confidentiality and user privacy, but query assurance, in the ODBS model with dynamic tree-indexed data supports. In the next sections we will elaborate on these techniques and extend them in order to deal with the three security objectives of query assurance as mentioned above.

Contrary to user privacy, although there are initial research activities (Gertner, Ishai, Kushilevitz, & Malkin, 1998; Du & Atallah, 2000; Dang, 2006b), the problem of data privacy still needs much more attention. Gertner et al. (1998) first considered the data privacy issue in the context of PIR-like protocols and proposed the so-called SPIR, symmetrical PIR, protocol in order to prevent users from knowing more than the answers to their queries. Unfortunately, such PIR-based approaches cannot be applied to the ODBS model because the data owners in PIR-like protocols are themselves the database service providers. Du and Atallah (2000) introduced protocols for secure remote database access with approximate matching with respect to four different ODBS models requiring different security objectives among those presented in the previous section. Even so, their work did not support outsourced tree-indexed data. In our recent work (Dang, 2006b), we presented a solution to ensuring data privacy in the ODBS model which can be applied to tree-indexed data as well. Nevertheless, our proposed solution must resort to a *trusted* third party, which is not easy to find in practice.

Recently, addressing the three issues of query assurance has also attracted many researchers and, as a result, a number of solutions have been proposed (e.g., Boneh et al., 2003; Mykletun et al., 2004; Pang & Tan, 2004; Pang et al., 2005; Narasimha & Tsudik, 2006; Sion, 2005; Nguyen, Dang Son, & Kueng, 2007). We must even now note that *none of them has given a solution to the problem of guaranteeing the query result freshness* (cf. Figure 2).[3] To prove the correctness of a user's query results, the state-of-the-art approaches (Boneh et al., 2003; Mykletun et al., 2004; Pang & Tan, 2004; Sion, 2005) employed some aggregated/condensed digital signature scheme to reduce the communication and computation costs. First, Boneh et al. (2003) introduced an interesting aggregated signature scheme that allows aggregation of multiple signers' signatures generated from different messages into one short signature based on elliptic curves and bilinear mappings. This scheme was built based on a "Gap Diffie-Hellman" group where the Decisional Diffie-Hellman problem is easy while the Computational Diffie-Hellman problem is hard (Joux & Nguyen, 2001). Despite the big advantage that this scheme can be applied to different ODBS models, it must bear a disadvantage related to the performance. As shown in Mykletun et al. (2004), the computational complexity of Boneh et al.'s (2003) scheme is quite high for practical uses in many cases. Second, Mykletun et al. (2004) introduced a RSA-based condensed digital signature scheme that can be used for ensuring authenticity and data integrity in the ODBS model. Their scheme is concisely summarized as follows.

Condensed-RSA Digital Signature Scheme

Suppose pk=(n, e) and sk=(n, d) are the public and private keys, respectively, of the RSA signature scheme, where n is a k-bit modulus formed as the product of two k/2-bit primes p

and q. Assume $\varphi(n)$=(p-1)(q-1), both public and private exponents e, d$\in Z_n^*$ and must satisfy ed $\equiv$ 1 mod $\varphi(n)$. Given t different messages $\{m_1, ..., m_t\}$ and their corresponding signatures $\{s_1, ..., s_t\}$ that are generated by the *same* signer, a condensed-RSA signature is computed as follows: $s_{1,t}=\prod_{i=1}^{t} s_i$ mod n. This signature is of the same size as a single standard RSA signature. To verify the correctness of t received messages, the user must multiply the hashes of all t messages and check that $(s_{1,t})^e \equiv \prod_{i=1}^{t} h(m_i)$ mod n.

As we can see, the above scheme is possible due to the fact that RSA is *multiplicatively homomorphic*. We will apply this scheme to our ODBS model in order to provide correctness guarantees of the received tree nodes from the server (cf. the section on "Correctness Guarantees"). Note that, however, this scheme is applicable only for a single signer's signatures. Sion (2005) also employed this approach to deal with the correctness of query results in his scheme. Besides, Pang and Tan (2004) applied and modified the idea of "Merkle Hash Trees" (MHT) (Merkle, 1980) to provide a proof of correctness for edge computing applications, where a trusted central server outsources parts of the database to proxy servers located at the edge of the network. In Narasimha and Tsudik (2006), however, the authors pointed out possible security flaws in this approach.

Furthermore, there are also some approaches to deal with the completeness of a user's query results (Sion, 2005; Pang et al., 2005; Narasimha & Tsudik, 2006). First, Sion (2005) proposed a solution to provide such assurances for arbitrary queries in outsourced database frameworks. The solution is built around a mechanism of runtime query "proofs" in a challenge-response protocol. More concretely, before outsourcing the data, the data owner partitions its data into k segments $\{S_1, ..., S_k\}$, computes hashes for each segment, $H(S_i)$, $i=\overline{1,k}$, then stores (outsources) them all together at the service provider's server. In addition, the data owner also calculates some "challenge tokens" with respect to S_i. Actually, the challenge tokens are queries that the data owner already knows their results, which can be used for verification later. Whenever a *batch* of queries are sent to the server, certain challenge token(s) are also sent together. The result set is then verified using the challenge tokens for its completeness. Although this approach can be applied to different query types, not 100% of the query assurance (the completeness) can be guaranteed because there are chances for a malicious server to "get away" with cheating in the query execution phase (i.e., the server only needs to "guess" and return the correct answer to the challenge token together with fake result sets for other queries in the batch, but nothing else). Moreover, this approach also introduces cost inefficiency for database updates because the challenging answers must be recalculated. Seriously, although the author did not aim to address the user privacy issue in the article, we should note that user privacy in this approach may be compromised because the server knows what data segments are required by the user so *inference and linking attacks* can be conducted (Dang, 2006b; Damiani et al., 2003). Second, Pang et al. (2005) introduced a solution based on aggregated signature schemes and MHT to provide the completeness of the query result. This approach is an extension of that presented in their previous work (Pang & Tan, 2004), which has been proven insecure due to some possible security flaws (Narasimha & Tsudik, 2006). Last, Narasimha and Tsudik (2006) developed an approach, called digital signature aggregation and chaining (DSAC), which achieves both correctness and completeness of query replies. However, in their approach, tuples must be pre-sorted in ascending order with respect to each searchable dimension for calculation of the signature chain, and thus it still does not support outsourced tree-indexed data where the order of tree nodes' contents is not able to be determined. This pre-sorting requirement also has a negatively tremendous impact on data updates, hence the total performance of the system will be degenerated.

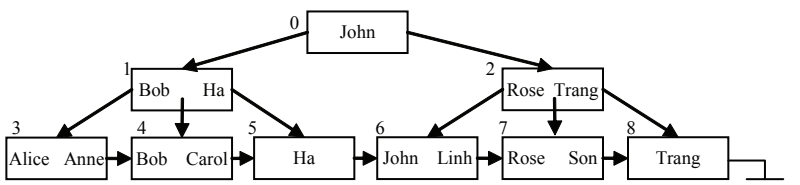
Apart from the security issues mentioned above and in the previous section, as we can observe in Figure 2, there exists another question: How can the server conduct *auditing activities*

in systems provided with such security guarantees (without employing special hardware equipment)? The server may not know who is accessing the system (e.g., Dang, 2006b), what they are asking for, what the system returns to the user, and thus how it can effectively and efficiently tackle the accountability or develop intrusion detection/prevention systems. The goals of privacy-preserving and accountability appear to be in contradiction, and an efficient solution to balance the two is still open. More discussions about this topic can be found in Burmester, Desmedt, Wright, and Yasinsac (2004). In the next section we will elaborate on the state-of-the-art approach proposed in Dang (2006a) to managing the storage and retrieval of dynamic outsourced tree-indexed data, and in the section after that we will extend this approach to strengthen it with query assurance supports, including all the three concerned security objectives.

A PRAGMATIC APPROACH TO MANAGING OUTSOURCED TREE-INDEXED DATA

As discussed in the literature, tree-based index structures take an indispensable role in both traditional and modern database applications (Dang, 2003). However, in spite of their advantages these index structures introduce a variety of difficulties in the ODBS model (Du & Atallah, 2000; Dang, 2006b). To detail the problem, Figure 3a illustrates an example of the B^+-tree for an attribute *CustomerName* with sample values. All tree nodes were encrypted before being stored at the outsourcing server to ensure the data confidentiality. Assume that a user is querying all customers whose name is *Ha* on this tree. If we do not have a secure mechanism for the query processing, a sequence of queries that will access in sequence nodes 0, 1, and 5 with respect to the above query will be

Figure 3. An example of the B^+-tree (a) and the corresponding plaintext and encrypted table (b)



(a) B+Table

NID	Node
0	(1,John,2,-,-1)
1	(3,Bob,4,Ha,5)
2	(6,Rose,7,Trang,8)
3	(Alice,Anne,4)
4	(Bob,Carol,5)
5	(Ha,-,6)
6	(John,Linh,7)
7	(Rose,Son,8)
8	(Trang,-,-1)

NID	EncryptedNode
0	D0a1n2g3Kh75nhs&
1	T9&8ra§ÖÄajh³q91
2	H&$uye”µnÜis57ß@
3	L?{inh*ß²³&§gnaD
4	Wh09a/[%?Ö*#Aj2k
5	j8Hß}[aHo$§angµG
6	#Xyi29?ß~R@€>Kh
7	~B³!jKDÖbd0K3}%§
8	T-§µran&gU19=75m

(b) B+EncryptedTable

revealed to the server. In addition, the server also realizes that the user was accessing nodes 0, 1, and 5, and node 0 is the root, node 1 is an internal node, node 5 is a leaf node of the tree, and so the user privacy is compromised. More seriously, using such information collected gradually, together with statistical methods, data mining techniques, and so forth, the server can rebuild the whole tree structure and infer sensitive information from the encrypted database, hence data confidentiality can also be spoiled. Besides, during the querying, the user will also get more information showing that there are at least two other customers named *John* and *Bob* in the database, so the data privacy is not satisfied (note that, however, we will not address the data privacy problem in this article).

Although Damiani et al. (2003) proposed an approach to outsourced tree-based index structures, it unfortunately has some security flaws that may compromise the desired security objectives (Dang, 2005, 2006b). Lin and Candan (2004) and Dang (2006a) developed algorithms based on access redundancy and node-swapping techniques to address security issues of outsourced tree-indexed data. We briefly summarize their solutions in the rest of this section.

Obviously, as private data is outsourced with search trees, the tree structure and data should all be confidential. As shown in Damiani et al. (2003), encrypting each tree node as a whole is preferable because protecting a tree-based index by encrypting each of its fields would disclose to the server the ordering relationship between the index values. Lin and Candan's (2004) approach also follows this solution and, like others (Dang, 2005, 2006a; Damiani et al., 2003), the unit of storage and access in their approach is also a tree node. Each node is identified by a unique *node identifier* (NID). The original tree is then stored at the server as a table with two attributes: NID and an encrypted value representing the node content. Consider an example: Figure 3a shows a B^+-tree built on an attribute *CustomerName*; Figure 3b shows the corresponding plaintext and encrypted table used to store the B^+-tree at the external server. As we can see, that B^+-tree is stored at the external server as a table over schema B^+EncryptedTable = {NID, EncryptedNode}.

Based on the above settings, Lin and Candan (2004) proposed an approach to oblivious traversal of outsourced search trees using two adjustable techniques: *access redundancy* and *node swapping*.

Access Redundancy

Whenever a user accesses a node, called the target node, it asks for a set of m-1 randomly selected nodes in addition to the target node from the server. Hence, the probability that the server can guess the target node is 1/m. This technique is different from those presented in Damiani et al. (2003), where only the target node is retrieved (this may lead to reveal the tree structure as shown in Dang, 2005, 2006b). Besides the access redundancy, it also bears another weakness: it can leak information on the target node position. This is easy to observe: multiple access requests for the root node will reveal its position by simply calculating the intersection of the redundancy sets of the requests. If the root position is disclosed, there is a high risk that its child nodes (and also the whole tree structure) may be exposed (Lin & Candan, 2004). This deficiency is overcome by secretly changing the target node's address after each time it is accessed.

Node Swapping

Each time a user requests to access a node from the server, it asks for a *redundancy set* of m nodes consisting of at least one empty node along with the target one. The user then: (1) decrypts the target node, (2) manipulates its data, (3) swaps it with the empty node, and (4) re-encrypts all m nodes and writes them back to the server. Note that this technique must re-encrypt nodes using a different encryption scheme/key (see Lin & Candan, 2004, for details). The authors proved that, with this technique, the possible position of the target node is randomly distributed over the data storage space at the untrusted server,

and thus the weakness of the access redundancy technique is overcome.

Additional Procedures

To realize oblivious traversal of outsourced search trees, some more critical issues must also be addressed:

- **Managing root node address:** The authors proposed to employ a special entry node called SNODE whose NID and decryption key are known to all valid users. It keeps pointers ROOTS pointing to the root nodes of all outsourced search trees that the user can access.
- **Managing empty node lists:** Empty nodes are stored in hidden linked lists. To help users find out the empty nodes, two other types of pointers are also stored in the SNODE: EHEADS and ETAILS point to the heads and tails of empty node lists, respectively.
- **Random choice for redundancy sets:** A redundancy set consists of the target node, an empty node, and m-2 randomly selected nodes. To enable users to do this, the SNODE records the range of NIDs of nodes in the data storage space at the server. The user will then be able to generate m-2 random NIDs within the range.
- **Managing the tree structure integrity:** This aims to maintain node/parent-node relationships after the node swapping. The first solution is to find the empty node to be swapped with the child node and update the parent node accordingly before actually swapping the child node. The second solution is to let users keep track of all nodes from the root down, deferring all the swaps until the node containing the data is accessed.
- **Concurrency control in the multi-user environment:** The authors also presented a solution to concurrency control without deadlocks. The main idea of the proposed solution is to organize nodes in the data storage space at the server into d levels, and each level requires an empty node list to store empty nodes at this level. Besides, all users access nodes in some fixed predetermined order, ensuring deadlock-free access in a multi-user environment. See Lin and Candan (2004) for detailed discussions about the proposed solutions to all of these critical issues.

Although Lin and Candan's (2004) approach only supports *oblivious* tree search operations, the two above techniques have served as the basis for our further investigation. Based on the access redundancy and node-swapping techniques, in Dang (2006a) we developed practical algorithms for privacy-preserving search, insert, delete, and modify operations that can be applied to a variety of *dynamic* outsourced tree-based index structures and the *unified user* model where data owners are also sole users of the system (see Mykletun et al., 2004; Du & Atallah, 2000; Dang, 2006b, for more details about ODBS models). Although our previous work provided the vanguard solutions for this problem with sound empirical results, it did not consider the query assurance problem. In the next section we will extend our previous work to address this problem. Note that, however, as with the unified user model, it is not necessary to take into account, in the remainder of this article, the clients as shown in Figure 1. The key "actors" in our concerned ODBS model now consist only of *the data owners* and *the outsourcing database server.*

QUERY ASSURANCE FOR OUTSOURCED TREE-INDEXED DATA

In this section, we present an extension of our previous work in Dang (2006a), which introduced solutions to the problems of data confidentiality and user privacy in the ODBS model (with respect to the unified user model), in order to incorporate solutions to ensuring the correctness, completeness, and freshness of the query results. The next section will detail the experimental results with real datasets.

Correctness Guarantees

As introduced in the first section, to guarantee the correctness of the query result set, the system must provide a means for the user to verify that the received data originated from the data owner as it is. As analyzed in previous sections, the state of the art employed the public key cryptography scheme to deal with this problem. With respect to our concerned ODBS model, where data privacy considerations are omitted and only single signer (i.e., only one data owner or multiple data owners using the same signature scheme) participates in the query processing, the RSA-based signature scheme is the most suitable as already discussed.

In our context, outsourced tree-indexed data is stored at the server side as described in the previous section—that is, as a table over schema EncryptedTable = {NID, EncryptedNode}. Before outsourcing the data, the data owner computes the hash h(m) of each encrypted node m. Here, h() denotes a cryptographically strong hash function (e.g., SHA-1). The data owner then "signs" that encrypted node m by encrypting h(m) with its private/secret key sk and stores the signatures together with EncryptedTable at the server. The table schema stored at the server therefore becomes *EncryptedTable = {NID, EncryptedNode, Signature}* (see Figure 4).[4] With these settings, users[5] can then verify each returned node using the data owner public key pk, hence ensuring the correctness of the result set.

Although the *naive* approach above ensures the security objective, it is expensive because the number of signatures to verify equals the redundancy set size. To solve this issue, we employ the condensed-RSA digital signature scheme based on the fact that RSA is multiplicatively homomorphic as presented in the previous section as follows: Given t input encrypted nodes $\{m_1, ..., m_t\}$ (the redundancy set) and their corresponding signatures $\{s_1, ..., s_t\}$, the server computes a condensed-RSA signature $s_{1,t}$ as the product of these individual signatures and sends it together with the redundancy set to the user. The user, in turn, will be able to verify the condensed signature $s_{1,t}$ by employing the hashes computed from all received nodes (in the corresponding redundancy set) as shown in the previous section. With this method, not only the query result correctness is ensured, but both communication and computation costs are also tremendously reduced. Note that in this case the server has to send only one condensed-RSA signature $s_{1,t}$ to the user for verification instead of t individual ones. In the following section we will show the experimental results.

Completeness Guarantees

Completeness guarantees mean that the server did not omit any tuples matching the query conditions. In our context, as a user asks the server for a redundancy set A of t nodes $A=\{m_1, ..., m_t\}$ and the server returns him a set R of t nodes $R=\{n_1, ..., n_t\}$, the user must be able to

Figure 4. EncryptedTable with tree node contents' signatures

B+Table

NID	Node
0	(1,John,2,-,-1)
1	(3,Bob,4,Ha,5)
2	(6,Rose,7,Trang,8)
3	(Alice,Anne,4)
4	(Bob,Carol,5)
5	(Ha,-,6)
6	(John,Linh,7)
7	(Rose,Son,8)
8	(Trang,-,-1)

B+EncryptedTable

NID	EncryptedNode	Signature
0	D0a1n2g3Kh75nhs&	s0
1	T9&8ra§ÖÄajh³q91	s1
2	H&$uye"µnÜis57ß@	s2
3	L?{inh*ß²³&§gnaD	s3
4	Wh09a/[%?Ö*#Aj2k	s4
5	j8Hß}[aHo$§angµG	s5
6	#Xyi29?ß~R@€>Kh	s6
7	~B³¹jKDÖbd0K3}%§	s7
8	T-§µran&gU19=75m	s8

verify that A=R. As presented in the previous section, a user asks for any encrypted nodes (at the server side) through their NIDs. Therefore, the user should be provided with a means of verifying that NID of each m_i, $i=\overline{1,t}$, equals NID of each corresponding n_i, $i=\overline{1,t}$. To ensure this, our solution is embarrassingly simple: an NID is encrypted with the corresponding node contents and this encrypted value is stored at the server side, together with its signature. Users can then check if the server returned the NIDs (in the redundancy set) that he or she did require (the completeness) as well as verify the query result correctness (as shown in the section, "Correctness Guarantees"). This idea is clearly illustrated in Figure 5.

In more detail, Figure 5 sketches settings for verifying completeness (and correctness) guarantees of the system. First, the encrypted value with respect to the attribute *EncryptedNode* also includes the NID of its corresponding node (for example, in the first row, the encrypted value also includes value 0). Second, the data owner signs each encrypted node using the RSA signature scheme, then stores the signature (e.g., s_0) together with the NID and its corresponding encrypted value as described in the previous section. Note that, however, verifying the completeness and the correctness must be carried out together—that is, the user cannot omit any of them and still be ensured that the other is also guaranteed. This is also true for freshness guarantees that we will present below.

Freshness Guarantees

As discussed previously, with *dynamic* outsourced databases, ensuring only the correctness and completeness of the result set is not enough. Apart from those, the system must also provide a means for users to verify that the received nodes are from the most recent database state, not the older one(s). Either motivating by clear cost incentives for dishonest behavior or due to intrusions/viruses, the server may return users *obsolete* nodes, which do not truly reflect the state of the outsourced database at the querying time. This is not a less important problem that also needs to be sorted out to make the ODBS model viable. Narasimha and Tsudik (2006) mention this problem and outline a possible solution based on MHTs, but no cost evaluation is given. Note that MHT-based approaches to the ODBS model are quite expensive, especially for *dynamic* outsourced tree-indexed data (Narasimha & Tsudik, 2006). In this section, we propose a vanguard solution to this problem, and a comprehensive evaluation for all concerned security objectives will be presented in the next section.

To solve the problem of freshness guarantees, users must be able to verify that the server did return them the most up-to-date required tree nodes (at the time it processed the query). Our solution is also quite simple, but sound and complete, based on timestamps: A timestamp of each child node is stored at its parent node.

Figure 5. Settings for verifying completeness guarantees

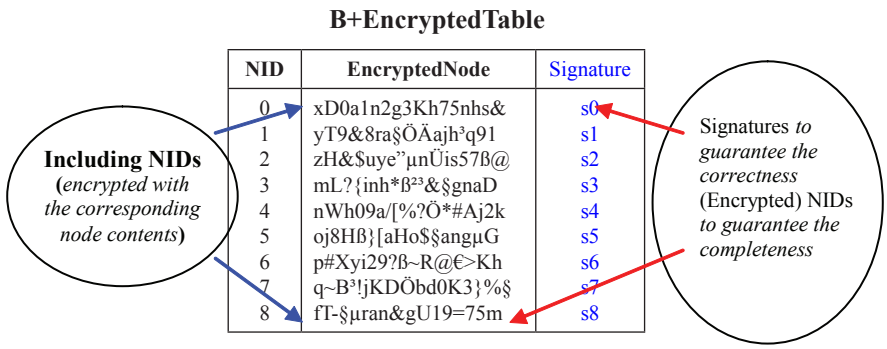
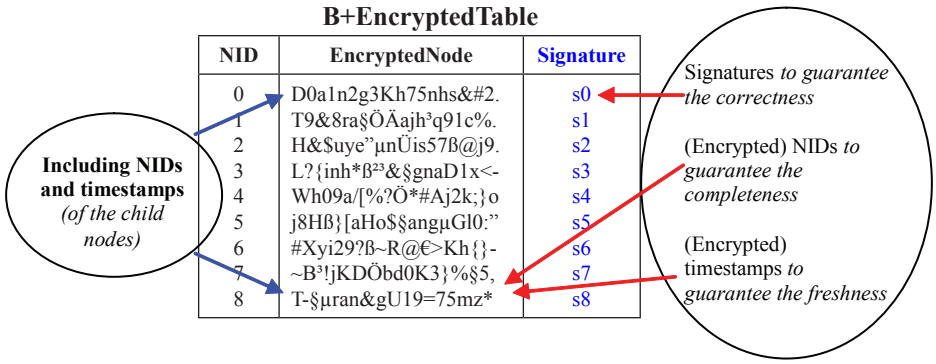


Figure 6. Settings for verifying freshness guarantees



This timestamp changes *only* as the child node *contents* (but not its address) are updated. In other words, a node keeps timestamps of all of its child nodes, and a user can then check (from the root node) if the server returned him the latest version of the required node as follows: In accessing the root, the user knows in advance all timestamps of its child nodes, and as a child node is returned, he or she can check if this node's timestamp equals the known value, and so on. This process is carried out for every tree node required by the user.

There is, however, one question that arises: How can users check the root's timestamp? The answer to this question is not complicated: In the settings for access redundancy and node-swapping techniques, there is a special node called SNODE that keeps some metadata and the root's address (cf. the previous section). The SNODE's address and its decryption key are known to all qualified users. Besides, in the context of our concerned ODBS model, only data owners are able to make changes to, as well as to query, their outsourced data. Therefore, for the sake of freshness guarantees, SNODE will keep the timestamp of the root in addition to other information as mentioned before (this timestamp changes only as *the root contents* are updated by a data owner), and each qualified user (i.e., other data owners) is informed about the timestamp of SNODE (by the data owner who made the changes[6]). Moreover, with the settings for access redundancy and node-swapping techniques, besides the root's address and timestamp, the users must also use *other metadata in the SNODE* for performing operations on the tree (see Dang, 2006a, for detailed discussions about the outsourced tree operations). Hence, it is vitally important that the SNODE's timestamp is communicated to all users as discussed above to ensure the freshness of the result set.

Overall, with all of these above settings, all the three security objectives of the query assurance problem in the concerned ODBS model—that is, the correctness, completeness, and freshness guarantees of the query result—can be effectively verified. Note that the encrypted value representing the corresponding node contents now includes not only its NID, but also timestamps of the child nodes. The corresponding signature is computed based on this final encrypted value. This is clearly illustrated in Figure 6.

EXPERIMENTAL RESULTS

To confirm theoretical analyses carried out in previous sections and establish the practical applicability of our approach, we implemented a prototype system and evaluated the proposed solutions with real datasets. For all experiments, we used two-dimensional datasets, which were extracted from the SEQUOIA dataset at *http://www.rtreeportal.org/spatial.html*. The

SEQUOIA dataset consists of two-dimensional points of the format (x, y), representing locations of 62,556 California place names. We extracted five sub-datasets of 10K, 20K, 30K, 40K, and 50K points from the SEQUOIA dataset for experiments.

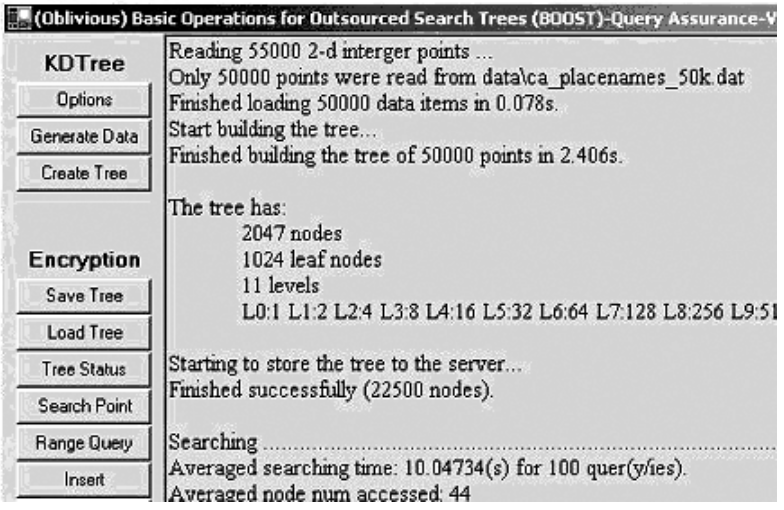
To manage the spatial points, we employed two-dimensional kd-trees due to its simplicity. For all the trees, we set the maximum number M of data items that a leaf node can keep to 50 and the *minimum fill factor* value to 4%. This means that each tree leaf node must contain at least two points and can store up to 50 points. Furthermore, the tree was stored in a data storage space with 22,500-node capacity (cf. Figure 7), divided into 15 levels of 1,500 nodes each (see Dang, 2006a; Lin & Candan, 2004, for detailed meanings of these settings).

Our prototype system consisted of only one P4 CPU 2.8GHz/1GB RAM PC running a Windows 2003 server. Both user and server were accommodated in the same computer, so for all experiments, we will report average time to complete a user request, which can represent the average CPU cost of each user request and analyze averaged IO and communication cost. In addition, all programs were implemented using C#/Visual Studio .NET 2003, and we employed the DES algorithm for the encryption of data and the RSA signature scheme (1024 bits key) with SHA-1 hashing for the digital signatures. We did experiments with all major basic operations, including search (for both point and range queries) and updates (inserts and deletes). Note that *modify* operations are combinations of inserts and deletes (Dang, 2005, 2006a).

In addition, because there is no previous work built on the same or similar scheme and addressing the same problem, we had to build our scheme from scratch and did experiments to evaluate our solutions to the query assurance issue on the basis of the condensed-RSA signature scheme and the naive/standard RSA signature scheme. All the security objectives of the query assurance issue (i.e., correctness, completeness, and freshness guarantees) were taken into account. The details are as follows.

Initially, we did experiments with the largest dataset, 50K points, for insert, delete, point, and range queries in order to see the performance of both naive and condensed RSA-based solutions. The *redundancy set* size is set to 4 for the tests. Figure 7 shows a screen

Figure 7. A screen shot: Costs of point and range queries with condensed-RSA scheme wrt. 50K points dataset



shot as we built the tree, stored it to the server, and performed point and range queries with condensed-RSA scheme wrt. the largest dataset. In Figure 8, we present the experimental results concerning the CPU cost for all operations. It is clearly shown that the condensed-RSA scheme CPU cost is much better than that of the naïve-RSA scheme. Note that the average accessed node number (i.e., the IO cost) of the two is the same, but the communication cost of the condensed-RSA scheme is also better by a factor of *(Redundancy_set_size -1) * RSA_signature_size.* This is due to the fact that as with the condensed-RSA scheme, the server has to send the user only *one condensed signature,* while it has to send *Redundancy_set_size* signatures with respect to the naïve-RSA scheme. Verifying more signatures is the main reason for a higher CPU cost of the latter.

Furthermore, to see the effect of different database sizes on the performance for each subdataset, we ran 100 separate queries with the redundancy set size being set to 4, and calculated averaged values for CPU time. With inserts, deletes, and point queries, we randomly chose 100 points from the corresponding dataset as the queries. With range queries, we randomly chose 100 squares as the queries. The sides of each square were chosen to be 1% of the norm of the data space side (if the dataset is uniformly distributed, this value maintains the selectivity of 0.01% for these range queries).The experimental results are shown in Figure 9. As we can see, the CPU cost saving of all kinds of queries is high, over 30% at the minimum between the condensed-RSA scheme and the naïve-RSA scheme. Again, as mentioned above, although the average accessed node number is equal for both schemes, the communication cost of the condensed-RSA scheme is better than that of the naïve-RSA scheme.

Security Analysis Briefings:

The security correctness of our proposed solutions to all the three security objectives in our holistic approach to the query assurance problem in the ODBS model is obvious and can be understood from discussions and analyses in previous sections. However, one may question the security degree of access redundancy and node-swapping techniques introduced in Lin and Candan (2004) and their *modified versions* for dynamic outsourced search trees (Dang, 2005, 2006a). The proof of the security correctness is therefore focused on that of the previous work. Fortunately, Dang (2006a) and Lin and

Figure 8. Condensed RSA signature scheme vs. naive RSA signature scheme

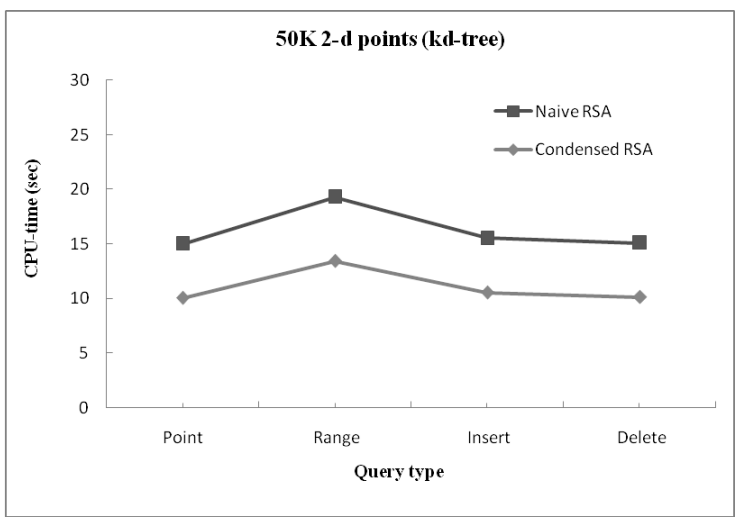
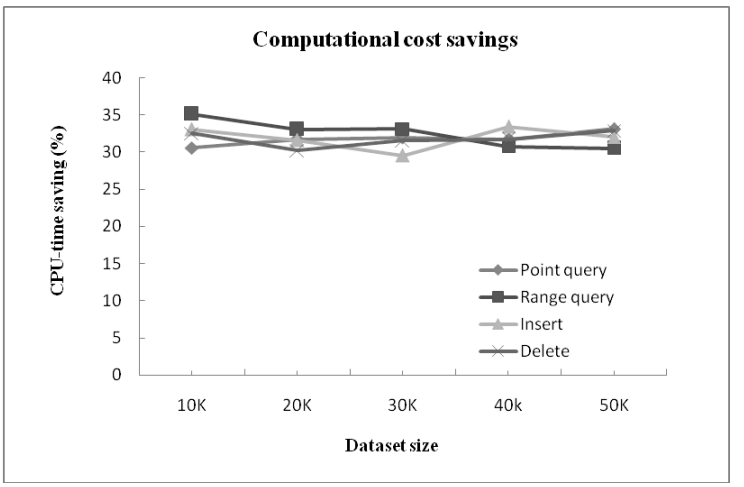


Figure 9. A variety of dataset sizes



Candan (2004) did prove that when users access the outsourced tree-indexed data, the untrusted server is not able to determine if any two queries are identical and, for two different queries, it is also not able to differentiate the distribution of the queries' redundancy sets in polynomial time. The former is to protect queries from the server and the latter aims to protect the tree structure. In summary, to conclude this section we emphasize that it has been mathematically proven in Lin and Candan (2004) and Dang (2006a) that our approach, based on the access redundancy and node-swapping techniques, is computationally secure to protect both queries and the tree structure from a polynomial time server. Hence, it is quite safe to claim that our proposed solutions in this article, which have extended the previous work, become full-fledged and can be applied to real-world ODBS models.

CONCLUSION AND FUTURE WORK

In this article, we explored the problem of query assurance in the oursourced database service (ODBS) model. Concretely, we extended our previous work (Dang, 2006a) and presented a full-fledged solution to the problem of ensuring the *correctness, completeness,* and *freshness* for basic operations (insert, delete, modify, point, and range queries) on dynamic outsourced tree-indexed data. Experimental results with real multidimensional datasets have confirmed the efficiency of our proposed solution. Notably, to the best of our knowledge, none of the previous work has dealt with all the three above security issues of query assurance in the ODBS model with respect to dynamic outsourced trees. Our work therefore provides a vanguard solution for this problem. Also, this work can also be applied to non-tree-indexed data outsourced to untrusted servers (with settings like those of Damiani et al., 2003; Dang, 2006a).

Our future work will focus on evaluating the efficiency of the proposed solutions in real-world applications and on addressing related open research issues. Specially, supporting multiple data owners' signatures (i.e., multiple signers) is a generalization of the proposed solution in this article. An efficient solution to this problem is still open. Moreover, as discussed in the section "Related Work," auditing and accountability for the ODBS model, as well as computer criminal-related issues and the data privacy problem, should be addressed, and they will be among our future research activities of great interest. Another problem

also attracts us: how to deal with the problem of *over redundancy* of the result set returned from the server—that is, the server sends the user more than what should be returned in the answers. This may cause a user to pay more for the communication cost, to incur higher computation costs, and so this issue needs to be investigated carefully.

ACKNOWLEDGMENTS

The author would like to thank the anonymous referees for their insightful reviews with interesting comments and suggestions.

REFERENCES

Asonov, D. (2001). Private information retrieval: An overview and current trends. *Proceedings of the ECDPvA Workshop* (pp. 889-894), Vienna, Austria.

Axelrod, C.W. (2004). *Outsourcing information security.* Norwood, MA: Artech House.

Burmester, M., Desmedt, Y., Wright, R.N., & Yasinsac, A. (2004). Accountable privacy. *Proceedings of the 12th International Workshop on Security Protocols* (pp. 83-95), Cambridge, UK.

Boneh, D., Gentry, C., Lynn, B., & Shacham, H. (2003). Aggregate and verifiably encrypted signatures from bilinear maps. *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 416-432), Warsaw, Poland.

Bouganim, L., & Pucheral, P. (2002). Chip-secured data access: Confidential data on untrusted servers. *Proceedings of the 28th International Conference on Very Large Data Bases* (pp. 131-142), Hong Kong.

Castano, S., Fugini, M.G., Martella, G., & Samarati, P. (1995). *Database security.* Boston: Addison-Wesley/ACM Press.

Chor, B., Goldreich, O., Kushilevitz, E., & Sudan, M. (1995). Private information retrieval. *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science* (pp. 41-50), Milwaukee, WI.

Chang, Y.-C., & Mitzenmacher, M. (2004). *Privacy preserving keyword searches on remote encrypted data.* Cryptology ePrint Archive Report 2004/051.

Dang, T.K. (2003). *Semantic based similarity searches in database systems: Multidimensional access methods, similarity search algorithms.* PhD Thesis, FAW-Institute, University of Linz, Austria.

Dang, T.K. (2005). Privacy-Preserving basic operations on outsourced search trees. *Proceedings of the IEEE International Workshop on Privacy Data Management,* Tokyo.

Dang, T.K. (2006a). A practical solution to supporting oblivious basic operations on dynamic outsourced search trees. *International Journal of Computer Systems Science and Engineering, 21*(1), 53-64.

Dang, T.K. (2006b). Security protocols for outsourcing database services. *Information and Security: An International Journal, 18,* 85-108.

Du, W., & Atallah, M.J. (2000). Protocols for secure remote database access with approximate matching. *Proceedings of the 7th ACM Conference on Computer and Communications Security, 1st Workshop on Security and Privacy in E-Commerce,* Greece.

Damiani, E., Vimercati, S.D.C., Jajodia, S., Paraboschi, S., & Samarati, P. (2003). Balancing confidentiality and efficiency in untrusted relational DBMSs. *Proceedings of the 10th ACM Conference on Computer and Communication Security* (pp. 93-102), Washington, DC.

Fong, K.C.K. (2003). *Potential security holes in Hacigümüs' scheme of executing SQL over encrypted data.* Retrieved from *http://www.cs.siu.edu/~kfong/research/database.pdf*

Gertner, Y., Ishai, Y., Kushilevitz, E., & Malkin, T. (1998). Protecting data privacy in private information retrieval schemes. *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (pp. 151-160).

Hacigümüs, H., Iyer, B.R., Li, C., & Mehrotra, S. (2002a). Executing SQL over encrypted data in the database-service-provider model. *Proceedings of the ACM SIGMOD Conference* (pp. 216-227), Madison, WI.

Hacigümüs, H., Mehrotra, S., & Iyer, B.R. (2002b). Providing database as a service. *Proceedings of the 18th International Conference on Data Engineering* (pp. 29-40), San Jose, CA.

Joux, A., & Nguyen, K. (2001). *Separating decision Diffie-Hellman from Diffie-Hellman in crypto-*

graphic groups. Cryptology ePrint Archive Report 2001/003.

Lin, P., & Candan, K.S. (2004). Hiding traversal of tree structured data from untrusted data stores. *Proceedings of the 2nd International Workshop on Security in Information Systems* (pp. 314-323), Porto, Portugal.

Merkle, R.C. (1980). Protocols for public keys cryptosystems. *Proceedings of the IEEE Symposium on Research in Security and Privacy* (pp. 122-134), Oakland, CA.

Mykletun, E., Narasimha, M., & Tsudik, G. (2004). Authentication and integrity in outsourced databases. *Proceedings of the 11th Annual Network and Distributed System Security Symposium,* San Diego, CA.

Narasimha, M., & Tsudik, G. (2006). Authentication of outsourced databases using signature aggregation and chaining. *Proceedings of the 11th International Conference on Database Systems for Advanced Applications* (pp. 420-436), Singapore.

Nguyen, V.H., Dang T.K., Son, N.T., & Kueng, J. (2007). Query assurance verification for dynamic outsourced XML databases. *Proceedings of the International Symposium on Frontiers in Availability, Reliability and Security (FARES)* (pp. 689-696), Vienna, Austria.

Pang, H.H., & Tan, K-L. (2004). Authenticating query results in edge computing. *Proceedings of the 20th International Conference on Data Engineering* (pp. 560-571), Boston.

Pang, H.H, Jain, A., Ramamritham, K., & Tan, K-L. (2005). Verifying completeness of relational query results in data publishing. *Proceedings of the SIGMOD Conference* (pp. 407-418).

PRIME Project. (2004). *The PRIME Project: Privacy and identity management for Europe.* Retrieved from *https://www.prime-project.eu*

Sion, R. (2005). Query execution assurance for outsourced databases. *Proceedings of the 31st International Conference on Very Large Data Bases* (pp. 601-612), Trondheim, Norway.

Smith, S.W., & Safford, D. (2001). Practical server privacy with secure coprocessors. *IBM Systems Journal, 40*(3), 683-695.

Thuraisingham, B. (2005). *Database and applications security: Integrating information security and data management.* Auerbach.

Umar, A. (2004). *Information security and auditing in the digital age: A managerial and practical perspective.* NGE Solutions.

ENDNOTES

1 In this article, however, we will deal with a less complicated ODBS model.

2 Note that, although privacy-related issues have been widely investigated (PRIME Project, 2004), the question "what is the complexity of the privacy problem?" is still open inasmuch as the answer is quite different, depending not only on technology, but also on sociology and politics (Thuraisingham, 2005).

3 In one of our very recent papers (Nguyen et al., 2007), which was written and published after this article had been accepted for publication, we also employed the timestamps as proposed in this article in order to ensure the query result freshness of dynamic outsourced XML databases.

4 Note that Figure 4 depicts only "real" tree nodes, but the empty nodes that are not shown here are also stored in the same table, EncryptedTable, over the same schema at the server side (see Dang, 2006a, for more information).

5 Here, in the unified user model, a user is also one of the data owners.

6 Obviously, the data owner knows who the valid users/other data owners are, and so this solution is well applicable.

Tran Khanh Dang received his BEng degree from the faculty of Computer Science & Engineering in HCMC University of Technology-HCMUT (Vietnam) in 1998. He achieved the medal awarded for the best graduation student. From 1998-2000, he had been working as a lecturer and researcher in the same faculty. Then, he got a PhD scholarship of the Austrian Exchange Service (OeAD) from 2000-2003, and finished his PhD degree (Dr.techn.) in May 2003 at FAW Institute, Johannes Kepler University of Linz (Austria). Afterwards, he had been working as a lecturer and researcher at the School of Computing Science, Middlesex University in London (UK) since August 2003. In October 2005, he returned home and has continued working for the Faculty of Computer Science & Engineering in HCMUT. Dr. Dang's research interests include database & information security, similarity search & flexible query answering systems, modern information systems & applications, and Grid data management. He has published more than 35 scientific papers in international journals & conferences. Dr. Dang has also participated in and managed many research as well as commercial projects.