Spatial Query Authentication using Merkle Quad Tree

J Linita Lyle[#1], Ani Sunny[*2]
[#]Department of Computer Science and Engineering, Mar Athanasius College of Engineering
Kothamangalam, Kerala, India.

Abstract

With the increasing popularity of location-based services and the excessive use of smart phones and GPS enabled devices, the practice of outsourcing spatial data to third party service providers has grown rapidly over the past few years. Meanwhile, the fast arising trend of cloud storage and cloud computing services has provided a flexible and cost-effective platform for hosting data from businesses and individuals, further enabling many location-based applications. However, in this database outsourcing paradigm, the authentication of the query results at the client remains a challenging problem. This paper presents a novel method for authentication of query results based on the ideas of Authentication Data Structures. The idea of Merkle Hash Tree is adopted which allows a client to verify the correctness and completeness of the result set. A novel authentication scheme is suggested based on the concepts of Merkle Hash Tree and quadtree for verifying the authenticity of outsourced spatial databases.

Keywords

Database Outsourcing, Security Authentication, Spatial Database.

I. Introduction

The embedding of geo-positioning capabilities (e.g., GPS) in mobile devices has sparked off several types of location-based services. Owing to the advancement of such technologies, it is becoming more convenient and motivating for mobile users to share with each other their experience with all kinds of points of interests (POIs) such as cafes, restaurants, tourist spots, worship places etc. In the GPS navigation system, a POI (point of interest) may be defined as a geographically anchored pushpin that someone may find useful or interesting, which is usually annotated with textual information (e.g., descriptions and users' reviews). In recent years various spatial query models and techniques have emerged to enhance user experiences of location based services.

Consider a scenario where a Data Owner (DO) possesses a proprietary spatial dataset, such as a specialized map overlay or a set of points of interest (e.g., local businesses). The data owner may be any non-governmental organization (NGO), or individuals. The dataset can be exploited to a profitable gain. However, the cost of setting up the infrastructure, hiring qualified personnel and advertising an online service may be impractical. Moreover, the dataset will be more valuable if it is combined with the functionality of a general-purpose online map. These reasons provide strong motivation for outsourcing the dataset to a specialized Location-Based Service Provider (LBSP). Due to advancements in cloud storage technology, network technology and improvements in data transmission techniques, database outsourcing has attracted the attention of organizations across the globe.

While analysing the security of the cloud storage, it can be assumed that the service provider, (LBSP) which maintains the outsourced spatial data, as untrusted. Clients must have the ability to verify the integrity of their own outsourced data. It can be assumed that the data owner is trusted, while the LBSP is untrusted. For example, the LBSP may replace some genuine results with others not among the top results or even not in the data owner's data set, and it may also modify some data records by adding more good reviews and features and deleting bad ones. In addition, an honest LBSP may be compromised by attackers to forge query results.

Query integrity is a line of research that ensures that a query result was indeed generated from the outsourced data (the authenticity requirement) and contains all the data satisfying the request (the correctness requirement). The key idea of the schemes used to ensure query integrity is that the data collector pre-computes and authenticates some auxiliary information (called authenticated hints) about its data set, which will be sold along with its data set to LBSPs. To faithfully answer a spatial query, a LBSP needs to return the correct spatial POI data records as well as proper authenticity and correctness proofs constructed from authenticated hints. The authenticity proof allows the query user to confirm that the query result only consists of authentic data records from the trusted data collector's data set, and the correctness proof enables the user to verify that the returned spatial POIs are the true ones satisfying the query.

II. BACKGROUND

Database outsourcing is closely associated with the concepts of Database as a Service (DBaaS).

According to technopedia, DBaaS can be defined as "A Cloud computing service model that provides users with some form of access to a database without the need for setting up physical hardware, installing software or configuring for performance."

By comparison, the DBaaS model is a fee-based subscription service in which the database runs on the service provider's physical infrastructure. Different service levels are usually available. In a classic DBaaS arrangement, the provider maintains the physical infrastructure and database, leaving the data owner to manage the database's contents and operation. Alternatively, a data owner can set up a managed hosting arrangement, in which the provider handles database maintenance and management. This latter option may be especially attractive to small businesses that have database needs, but lack the adequate IT expertise.

DBaas has various advantages which include: elimination of physical infrastructure, reduced cost, instantaneous scalability, performance guarantees, specialized expertise, fail over support and so on. However, outsourcing databases to third party services providers also poses some major security challenges which include confidentiality, integrity, availability, authenticity etc. Thus while outsourcing data to third party service providers, it is vital that efficient security mechanisms are incorporated to ensure that the database is not exploited for malicious intends by the service providers.

A. Architecture of Outsourced Database model

Figure 1 depicts the architecture of Outsourced Database Model. It consists of 3 main entities as

- Data owner
- Service provider
- Clients

Generally, data owner and end users are considered as trustful entities while service provider is distrustful in context of manipulating data in an unauthorized manner. A data owner uploads his/her data at third party service provider's site using high speed communication link. A data owner can manipulate his/her data which includes functions such as insertion, modification/updation, and deletion. In case of multiple clients, access level permissions can be set for using the data. The service provider stores the data uploaded by the data owner. Data management hardware and software tools are deployed and maintained at the provider's site. The service provider ensures availability of data and provides efficient mechanisms for end users to access or query the underlying data.

There are 3 types of outsourced database models, categorized on the basis of number of data owners and end users/clients involved. The first model is unified client model in which the database is used by a single entity i.e. the functionalities of the client and the data owner are the same. The data owner does all the operations on the database and the communication link between data owner and client has very high bandwidth.

The second type of outsourced database model is the multiple client model where multiple clients are given privileges of read only access. Here, the database can be accessed through mobile devices, laptops, PCs and the communication link has limited bandwidth.

The third type of model is the multiple data owner model. In this model, there might be more than one data owner outsourcing his/her database to a third party service provider. Hence, for every group of data owner and client, separate access control and security policies need to be applied. This model is also referred to as multi-authority outsourced database model.

III. LITERATURE REVIEW

Various security issues related to outsourcing databases to third party service provides have been identified, but the focus here is on the authentication mechanisms aimed at establishing correctness and completeness proofs of the outsourced data.

The idea of outsourcing databases to a third-party service provider was first introduced by Hacigumus et al. [7]. Since then, numerous query authentication solutions have been proposed for auditing query results in outsourced relational databases. In the recent years, with the growing popularity of location based services, spatial databases imposes additional challenges to the authentication process as the spatial distribution of the objects need to be considered as it plays a vital role in the querying process. Authentication Data structure based approaches are used widely and has proven to be more efficient than previously suggested approaches.

Authentication data structures are a model of computation where untrusted responders answer queries on a data structure on behalf of a trusted source and provide a proof of the validity of the answer to the user. Authentication data structure based approach are usually used for ensuring integrity and authenticity of outsourced databases.

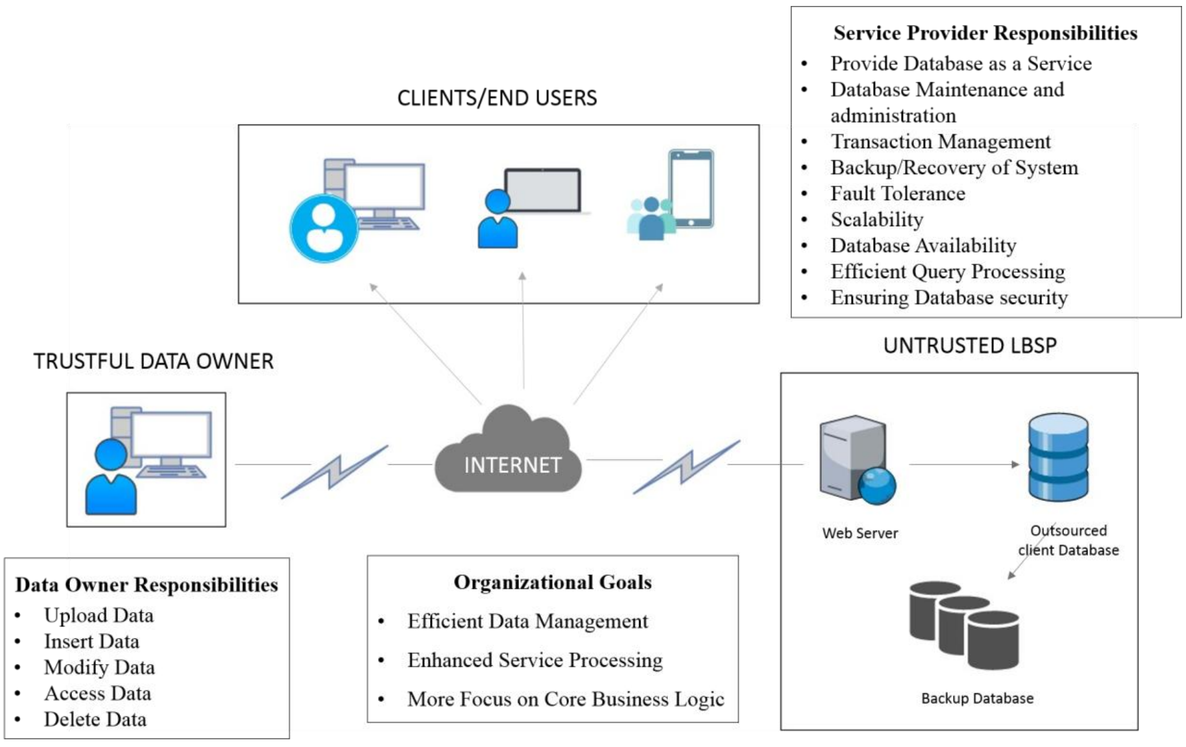
CLIENTS/END USERS
Service Provider Responsibilities
Provide Database as a Service
Database Maintenance and administration
Transaction Management
Backup/Recovery of System
Fault Tolerance
Scalability
Database Availability
Efficient Query Processing
Ensuring Database security
TRUSTFUL DATA OWNER
UNTRUSTED LBSP
INTERNET
Web Server
Outsourced client Database
Backup Database
Data Owner Responsibilities
Upload Data
Insert Data
Modify Data
Access Data
Delete Data
Organizational Goals
Efficient Data Management
Enhanced Service Processing
More Focus on Core Business Logic

Figure 1: Architecture of Outsourced Database Model

The most popular authentication data structure based approaches are MACs (Message Authentication Code), Digital Signature scheme, MHT (Merkle Hash Tree).

MAC function takes a secret key and a variable length data (message) as input and produces the MAC code. It assures integrity and authenticity in unified outsourced database model where data owner and client are the same entities and where bandwidth overhead and computation overhead are also less. MAC is not suitable in multiple clients and multiple data owner model. A deceitful client with secret MAC key can collude with the server and add fraudulent records to the database. Hence, non-repudiation cannot be assured with MACs.

Digital signatures are used to provide authenticity, integrity and non-repudiation and it is used when a large number of entities are involved in outsourcing [12]. A digital signature is a mathematical scheme for demonstrating the authenticity of digital messages or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity). Large storage and bandwidth is required for implementing the digital signature schemes.

MHT[11] is used for assuring the integrity, secure verification, authentication of large datasets. It works as follows: Suppose data value is represented as d1, d2… d_n. A leaf node Ni (i = 1, 2… n) stores the hashed value of data such that N1=h (d1) and so on. The non-leaf node is represented by concatenation of hash values of its children.

The combination of MHT and B-tree is implemented by [10] known as Merkle B-tree to provide integrity (completeness, correctness). In this, the client computes all the hashes of the sub-tree using the verification object in a repeated manner until the root of the tree. Once the hash of the tree is computed, the correctness can be verified using the owner's public key and hashed value of root. As the client is forced to find the hash of whole sub tree, it ensures the criteria of completeness. A dithered B-tree which is combination of original B-tree and its corresponding dithered B-tree is implemented in [5] based on key-value pairs. It prevents a third party from learning whether the key is present in database or not, thus ensuring privacy. As the server stores two kinds of trees, it requires more space. Also communication cost for transferring the page-level data is also more.

Nested Merkle B+ Tree which is an index structure is implemented by [15] to provide query assurance for dynamic outsourced XML databases. In this, all paths in XML document are listed out by tree traveller algorithm. The result contains leaf entries pointing to records, data records and not-in-result (co-path) entries. When the client fires query, server returns the root with signature and timestamp. Actual result and co-path are assembled in verification object (VO). The client recalculates hast of root and verifies it with the signature for assuring completeness and correctness.

In spatial databases, MHTs are suitable only for range queries and top K queries as the server returns only two full paths (Minimum value path and maximum value path) to the client.

MR Trees based on the idea of MHT and R* Trees is suggested in [15]. Leaf nodes are identical to those of the R* tree. Each entry R_i corresponds to a data object. Every leaf node of the tree stores a digest that is computed on the concatenation of the binary representation of all objects in the node. Internal nodes are assigned a digest that summarizes the child nodes' MBRs (minimum bounding rectangles) and digests. Digests are computed in bottom-up fashion and the single digest at the root is signed by the DO. The resulting *VO* contains all the objects in every leaf node visited, and the MBRs and digests of all the pruned nodes. With this information, the client can reconstruct the root digest and compare it against the one that was signed by the data owner. In addition, the client also examines the spatial relations between the query and each object/MBR included in the *VO*, in order to verify the correctness of the result.

An enhanced version of MR Trees, MIR Tree is suggested in [3, 6]. MIR is a combination of MHT and IR tree [18].The IR-tree is essentially an R-tree, each node of which is enriched with a reference to an inverted file for the objects contained in the subtree rooted at the node. In the IR-tree, a leaf node contains a number of entries of the form (O, O.r), where O refers to an object in database and O.r is the bounding rectangle of object O. A leaf node also contains a pointer to an inverted file for the text documents of the objects being indexed. The inverted file is stored separately from the R-tree, for two reasons: First, it is more efficient to store each inverted file contiguously, rather than as a sequence of blocks or pages that are scattered across a disk [18]. Second, the inverted file can be distributed across several machines, while this is not easily possible for the R-tree [14].

Various drawbacks have been identified in the structure of MR trees and in the verification process. First, the authentication information (hash digests) embedded in the MR-tree reduces the node fan out which results in increased I/O accesses during query processing. Second, in the presence of updates, all the digests on the path from an affected leaf node to the root have to be recomputed. When updates are frequent, query performance is degraded [8]. Finally, the overhead of the *VO* can be significant, especially for queries that return only a few objects. This is due to the fact that SP has to return all objects that lie inside the leaf nodes that are visited during query processing. An extension of the MR-tree, called MR*-tree [16], mitigates this last drawback by ordering the entries of each node and constructing hierarchical relationships of the digests. Nevertheless, it does not eliminate the VO overhead entirely, and it increases the verification cost at the client.

An approach based on voronoi diagrams was suggested in [9]. Voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane. The approached referred to as VN-Auth separates the authentication information from the spatial index, thus allowing efficient query processing at the service provider. Additionally, the verification information depends only on the object and its voronoi neighbours, and hence database updates can be disseminated quickly to their local regions and be performed independently of all other updates in the database. VN-Auth is proven to handle not only kNN and range queries, but also more advanced query types, such as reverse kNNs, k aggregate NNs and spatial skylines.VN-Auth produces compact verification objects, which enables fast query verification on mobile devices with limited capabilities.

IV. MERKLE QUADTREE FOR SPATIAL QUERY AUTHENTICATION

A. Quadtree

A quadtree is a space partitioning tree data structure in which a d-dimensional space is recursively subdivided into 2d regions. Due to its simplicity and regularity, the quadtree technique has been widely applied in many applications. As an efficient implementation of the disk-based quadtree, the linear quadtree is adopted to keep the non-empty leaf node of the quadtree in an auxiliary disk-based one dimensional structure (e.g., B+ tree), where each node can be encoded by the space filling curve techniques. Here, the quadtree nodes are encoded based on the Morton code (a.k.a. Z-order) because the Morton code of a node is encoded based on its split sequence, i.e., the path of the node in the quadtree, and the code of a particular node (region) in the space is unique.

The Morton code of a node based on its split sequence in 2-dimensional space can be derived in a simple and efficient manner. As shown in Figure 2, assuming that quadtrees resulting from a split are numbered in the order SW, SE, NW and NE, which are represented by 00, 01, 10 and 11 respectively. Then the code can be derived by concatenating the split codes in each subdivision.

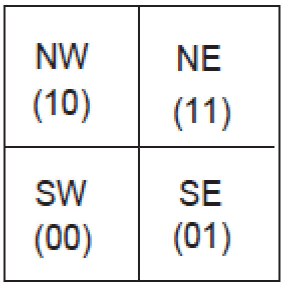

Figure 2: Morton code assignment

For example, Figure 3 shows the space partition and the corresponding tree structure of a simple quadtree for a given set of points p1. . . p4 where leaf nodes are labelled by their Morton codes.

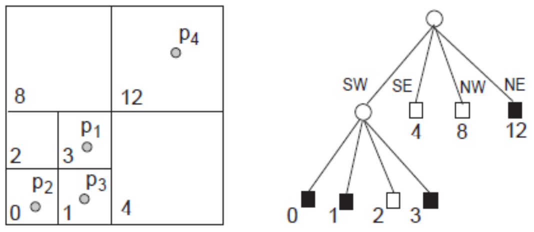

Figure 3: Space Partition and Quadtree generation

As shown in Figure 3, a circle and a square are used to denote the non-leaf node and leaf node respectively. Moreover, a leaf node is set black if it is not empty, i.e., it contains at least one point. Otherwise, it is a white leaf node. Suppose the maximal depth of the quadtree is 2, the split sequence of the node 1 is “SW, SE" and hence its code is represented by 0001. For the node at higher level, 0 is used to pad the remaining binary digits. For instance, the split sequence of the node 12 is “NE" and its code is represented by 1100 where the last two binary digits are padding.

The quadtree structure may be kept as the space partition based signature of the objects, and hence the level of a node in the quadtree is available during the query processing. Consequently, it is easier to come up with the correct node (region) based on the code and the level information.

For the linear quadtree, only the black leaf nodes on the disk by one dimensional index structure (e.g., B+ tree) are maintained, which are ordered by their Morton codes. Figure 4 shows an ordering results of these black leaf nodes as well as the objects resident on them, where the node codes are represented as integer numbers.

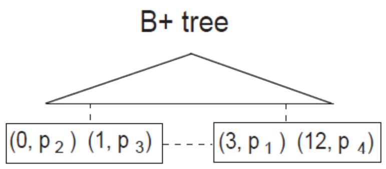

Figure 4: B+ Tree representation of a quadtree

B. MHT based on Quadtree for verifying authenticity

Quadtrees can be used to effectively locate points in space and hence when combined with the ideas of Merkle Hash Tree can be used to efficiently authenticate spatial databases.

The Merkle Quadtree tree is a tree structure where each node contains at most four children and each leaf node contains the hash of a data value (representing a point of interest), and each internal node contains the hash of the concatenation of its children. Verification of data values is based on the fact that the hash value of the root of the tree is authentically published (authenticity can be established by a digital signature). To prove the authenticity of any data value, in addition to the data value, the prover has to provide the verifier, the values stored in the siblings of the path that leads from the root of the tree to that value. The verifier, by iteratively computing all the appropriate hashes up the tree, can simply check if the hash computed for the root matches the authentically published value.

This method can be efficiently used to prove the existence of an element in the set stored among the leaf nodes. Clients issue queries and receive as query replies a set of tuples, or parts of tuples (selected attributes), that they wish to authenticate. Let n be the number of values in the database and t, the number of tuples in the result set returned to clients. Assume that data owner has built a Merkle Quadtree for a relation. The root has been signed by the owner and the MHT given to the untrusted outsourced database server. A client then wishes to query the server about the existence of a particular attribute value, v. If v is present in the tree it will be represented by one of the leaves and the server will return the nodes on the co-path from the specific leaf node up to the root. The nodes on the co-path are defined as those needed to enable the client to re-compute the root of the tree, such as to then verify its signature (which was generated by the data owner). In the event that the signature is valid, the client can be assured of that the response returned by the server is correct, and therefore that the attribute value indeed is present.
The traversal of the tree is usually done in a bottom-up approach. However, it is possible to allow for a top-down approach, which has the advantage of allowing the client to simulate a search in the tree and thereby avoiding the need of transferring any

structural information related to the reconstruction of the root. To accomplish this, some additional information needs to be present in each internal node, for example, the largest value in its left subtree, such as to help guide the search.

Another feature of the proposed method is the ability to give empty proofs, namely, prove that a value is non-existent in the tree, i.e., that a record is non-existent in the database. This is accomplished by returning the co-paths for the neighbor leaves covering the range of the selected record queried upon.

V. CONCLUSIONS

Authentication of outsourced data sets is an important aspect of security in outsourced databases, to ensure that the original database has not been tampered with. Various authentication mechanisms have emerged in the recent years. In this PAPER, AN authentication data structure based Merkle hash tree is adopted as an efficient authentication mechanism for checking the integrity of outsourced spatial databases. A novel authentication mechanism based on the ideas of Merkle Hash tree and quadtree is developed which is used to efficiently verify the authenticity of outsourced spatial databases. The novel approach yields significant performance in terms of computational overhead and restructuring the tree during updates when compared with the previous methods.

REFERENCES

[1] B. Hore, S. Mehrotra, and G. Tsudik, “A Privacy-Preserving Index for Range Queries,” Proc. 30th Int’l Conf. Very Large Data Bases (VLDB’04), pp. 720-731, Aug. 2004.

[2] Chung-Min Chen, Andrzej Cichocki, Allen McIntosh, Euthimios Panagos, “Privacy-Protecting Index for Outsourced Databases”, In Proc. of ICDE Workshops 2013, pp. 83-87.

[3] D. Wu, G. Cong, and C. S. Jensen. A framework for efficient spatial web object retrieval. VLDB J., 21(6):797–822, 2012.

[4] Dongxi Liu, Shenlu Wang, “Programmable Order-Preserving Secure Index for Encrypted Database Query” In Proceedings of 2012 IEEE Fifth International Conference on Cloud Computing, pp. 502-509, 2012.

[5] E. Mykletun, M. Narasimha, and G. Tsudik, “Authentication and *integrity* in outsourced databases,” In Proc. of ACM Trans. On Storage, vol. 2, 2006, pp. 107-138.

[6] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. In VLDB, pages 337–348, 2009.

[7] H. Hacigumus, S. Mehrotra, and B. Iyer, “Providing Database as a Service,” Proc. IEEE 18th Int’l Conf. Data Eng. (ICDE), Feb. 2002.

[8] H. Pang, J. Zhang, and K. Mouratidis. Scalable Verification for Outsourced Dynamic Databases. PVLDB, 2(1):802–813, 2009.

[9] Hu, L., Ku, W., Bakiras, S., Shahabi, C.: Spatial query integrity with voronoi neighbors. IEEE Trans. Knowl. Data Eng. 25(4), 863–876 (2013)

[10] Li Feifei, Marios H, George K, “Dynamic Authenticated Index Structures for Outsourced Database” In Proc. of ACM SIGMOD’06. Chicago, Illinois, 2006, pp. 121-132

[11] R. C. Merkle. A certified digital signature. In *CRYPTO*, pages 218–238, 1989.

[12] R. J. Morteza Noferesti, Mohammad Ali Hadavi, "A Signature-based Approach of Correctness Assurance in Data Outsourcing Scenarios," ICISS 2011, India, pp. 374-378.

[13] R. Zhang, J. Sun, Y. Zhang and C. Zhang, "Secure Spatial Top-k Query Processing via Untrusted Location-Based Service Providers," in IEEE Transactions on Dependable and Secure Computing, vol. 12, no. 1, pp. 111-124, Jan.-Feb. 2015.

[14] Schnitzer, B., Leutenegger, S.: Master-client R-trees: a new parallel R-tree architecture. In: SSDBM, pp. 68–77 (1999)

[15] Viet Hung Nguyen, Tran Khanh Dang, Nguyen Thanh Son , Josef Küng, “Query Assurance Verification for Dynamic Outsourced XML Databases”, Second International Conference on Availability, Reliability and Security (ARES'07), 2007, pp. 689 - 696.

[16] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios. Authenticated Indexing for Outsourced Spatial Databases. VLDB J., 18(3):631–648, 2009.

[17] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios. Spatial Outsourcing for Location-based Services. In ICDE, pages 1082– 1091, 2008.

[18] Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. 38(2), 1–56 (2006)