CERTIFYING DATA FROM MULTIPLE SOURCES

Glen Nuckolls
Department of Computer Science
University of Texas, Austin
nuckolls@cs.utexas.edu

Chip Martel
Department of Computer Science
University of California, Davis
martel@cs.ucdavis.edu

Stuart G. Stubblebine
Stubblebine Research Labs
8 Wayne Blvd., Madison, NJ 07940
stuart@stubblebine.com

Abstract

Data integrity can be problematic when integrating and organizing information from many sources. In this paper we describe efficient mechanisms that enable a group of data owners to contribute data sets to an untrusted third-party publisher, who then answers users' queries. Each owner gets a proof from the publisher that his data is properly represented, and each user gets a proof that the answer given to them is correct. This allows owners to be confident that their data is being properly represented and for users to be confident they are getting correct answers. We show that a group of data owners can efficiently certify that an untrusted third party publisher has computed the correct digest of the owners' collected data sets. Users can then verify that the answers they get from the publisher are the same as a fully trusted publisher would provide, or detect if they are not. The results presented support selection and range queries on multi-attribute data sets and are an extension of earlier work on Authentic Publication which assumed that a single trusted owner certified all of the data.

1. Introduction

Many settings require data from multiple sources to be combined into a single online database. A combined data set offers the convenience of a single source for queries and support for query types that are dependent on the overall data set. These benefits come at the cost of introducing new security concerns. Principal among them is the need to ensure the honesty of the party who collects the data and provides answers to users' queries. In order to assure accurate answers to queries, we need to prevent inadvertent and malicious data corruption.

In this paper we provide a formal framework and specific implementations which address the problems that arise when an untrusted third party publisher collects and organizes data from many different data owners and then provides answers to user queries on the combined data set. We focus on achieving the two closely related goals of providing a third-party publisher with **efficient** mechanisms to:

(1) Assure data owners that their data will be accurately represented in answers to user queries.

(2) Assure users that the answers to their queries are accurate.

MOTIVATION AND APPLICATIONS. Sites which provide provably accurate data have many important applications including consumer, financial, medical and government settings. Many of these applications draw from multiple sources for their content and, as the Internet expands, the need for accurate data will only increase. As data becomes more valuable, the motivation for dishonest data representation increases and thus the need for mechanisms to prevent it.

A product pricing engine provides a simple and familiar example where a number of data owners, here retailers, contribute data to an untrusted publisher, the pricing engine. If a user asks for a list of all digital cameras matching a given description, they want to know that they have the cheapest listing and that everything in the list is a legitimate. The retailers also want to be sure that the pricing engine is not excluding or misrepresenting their products in favor of other retailers.

In this situation, we want the retailer to be confident that the pricing engine is returning all of his products which match the users description. More generally, we want a retailer to be sure that all of his products that match the query are returned to the user. The user also should be confident that all of the returned data is from legitimate sources. We want to achieve these goals without forcing a retailer or consumer to spend large amounts of extra resources to ensure data integrity.

Our solutions provide mechanisms to support multi-attribute queries which expands the range of database applications.

EXTENDING AUTHENTIC PUBLICATION. There are many possible ways to assure data integrity. Our solutions are an extension of prior work on Authentic Publication [6] which allows untrusted third party publishers to provide users with query answers and short proofs that the answer given is correct. Merkle trees [13, 14] provided the original basis for this work. Since our results extend Authentic Publication to multiple owners, we briefly recap the basic construction. Our example uses a binary search tree, for a data set D, with the values stored at the leaves. The tree is digested using a cryptographic hash function as follows: Starting at the leaves, the digest value of a leaf is the hash of its data value. The digest value of an internal node is the hash of the digest values of its (two) children. The overall digest value of the tree is just the digest value at the root. With this digest value, an efficient proof, of size $O(\log |D|)$ can be given that a data item is or is not in the set.

Authentic Publication has this initial setup: (1) A trusted owner digests the data (e.g. with a binary search tree); (2) the data is given to one or more untrusted publishers; (3) the digest is distributed to the users. Users now send queries to an untrusted publisher, and the publisher returns an answer and a proof that the answer is the same as would be given by the trusted owner. Efficient protocols based on the Merkle tree approach [6] and general methods for producing them [12] exist for a wide range of query types and were proved to be secure in [12] as long as the publisher can't find a collision in the hash function. This approach has many advantages including scalability, since anyone can be a publisher, efficiency, since it uses efficient hash computations, and security, since there are no secret keys used in the proofs to be compromised.

MULTIPLE OWNER SETTING. We want to extend the advantages of Authentic Publication to the multiple owner setting. However, with no single trusted source and limited communication among the owners, there are new challenges. Now, not only does the publisher need to convince the user that the digest used in verification is a valid one, he must also convince each owner. One possible solution has the publisher give the entire collected data set to each owner, who then computes the digest on his own, more or less replicating the single owner Authentic Publication protocol. However, each owner must now deal with the entire database, and we want each owner's work to depend only on the size of his own data set. There are significant difficulties presented by this since each owner only sees a part of the data and digest structure being verified. Since the publisher is not trusted, we must also prevent him from creating and including his own values in the data set and digest. We now give a high level view of our protocols:

Consider a situation in which a set of n data owners $O_1, O_2, \ldots, O_n$ wish to combine their respective data sets $D_1, D_2, \ldots, D_n$ to support multiple queries for many users U. Assume that a third party publisher P collects and combines the sets, and provides the answers to users' queries. The publisher needs to convince each owner that his data set is correctly included and each user that her answer is accurate. A typical sequence of events is in figure 1 as follows:

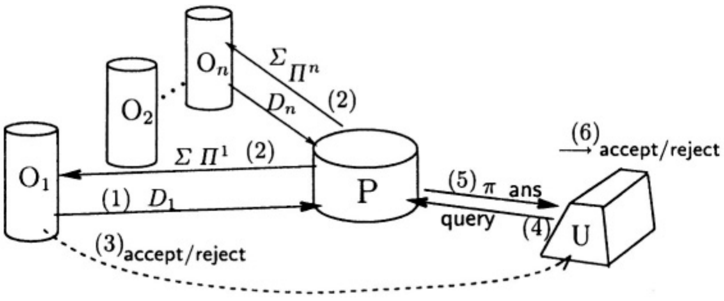

Figure 1. Typical protocol execution.

1 The owners O_i send their data sets D_i to the publisher P.

2 P collects the data sets and computes a digest Σ of the combined data set $D = \bigcup_{i=1}^{n} D_i$, and for each owner, a proof Π^i that the data set D_i is correctly included in Σ. Π^i and Σ are sent to each owner. Σ is also made available to the users.

3 Each owner evaluates his proof and either accepts or rejects, notifying any users of this result and the digest value Σ.

4 A user U sends query to P.

5 P computes an answer ans and a proof π that ans is correct, returning these to U.

6 U evaluates ans and π, and either accepts or rejects.

We assume that each owner and user receives the same value for Σ, whether or not it is accurate. We also assume that every party has access to the hash function H which is used to produce the digest and in the verification procedures. With multiple owners, we can either assume that all owners are trusted to complete the protocol, or that some owners are untrusted, and may conspire with the publisher or other untrusted owners. We outline the results in each case. We will also consider settings where a trusted third-party collects owner approvals of the digest Σ. This reduces the need for owner-user interactions, and for the user to know the details of which owner's contributed to the data set.

GOALS. When an untrusted party collects data for (re)publication there are a variety of guarantees users and data owners might desire. Perhaps the strongest is that the answer to any query is exactly the same as would have been given by a trusted data collector who correctly acquired the owners' data. In the Trusted owners section below we discuss achieving this strong result for multi-attribute queries.

An important, but weaker goal would be to assure an owner (and users) that all his data will be correctly included, but the answer might also include bad data from other, or non-existent owners (e.g. for a selection or range query). An additional goal is to assure the owner that no data items not in his data set will be falsely reported as his. We show this is possible for a broader range of settings.

TRUSTED OWNERS. For a collection of n trusted owners, we provide an efficient protocol that an untrusted publisher can use to compute a digest value Σ for a binary search tree storing the owners' combined data set. The publisher provides a proof to each owner, proportional in size to each owner's data set (and the log of the combined set), that Σ is valid. If each owner accepts their proof, and there is no collision found in the hash function, then Σ is exactly what a fully trusted publisher would have produced for the owners' combined data. We achieve this using a *count certified search tree* which we define and use to support the protocol. The count certified search tree supports, among other query types, authentication of answers to selection and range queries. Users can query the Publisher many times based on Σ, just as in the single owner Authentic Publication setting. We also extend this result to digests of multi-dimensional range trees which allows a richer set of multi-attribute queries (e.g. return all digital cameras which have 1-3 mega-pixels, cost \$250-\$400 and weigh at most 6 ounces). The results in this setting are given in section 2.

UNTRUSTED OWNERS. If some owners cannot be trusted to follow the protocol, and might be conspiring with the publisher, we still want to guarantee each honest owner that if he approves the digest of a publisher, then his data will be properly included in answers to queries. As before, each owner gets a proof of size proportional to his individual data set (and the log of the entire data set). If an owner approves this proof, then he can be confident that any answer a user accepts will contain exactly those data items from his data set which satisfy the user's query. This works for selection, range, and multi-dimensional range queries (e.g. in our digital camera example above, a retailer who approved the digest value of a publisher could be confident that all of his cameras which fit the user's criteria would be returned). We also provide new, efficient mechanisms to prevent false data attribution. An honest owner O can be sure that the publisher cannot return a data item which is claimed to belong to O, but is not in O's data set.

1.1. Background and Related Work

Data authentication has been used for time-stamping [9] and micro-payments [4]. The work in this paper is an extension of earlier work on Authentic Publication [6]. The techniques used in Authentic Publication are based on the original work by Merkle [13, 14] and refinements by Naor and Nissim [15] for certificate revocation.

Devanbu and Stubblebine showed how to create authenticated versions of stacks and queues in [7], and Maniatis and Baker [11, 10] present similar methods for authenticating B-trees and using them for secure data repositories. In [6], we introduced the general idea of authentic data publication, and showed how to securely answer a broader range of queries. This work is extended to authentic publication of XML documents in [5] using authenticated tries, and in [12] we introduced a general model and framework for Authentic Publication for a wide range of query types by defining a general class of authenticated data structures. Goodrich, Tamassia, Triandopoulos and Cohen applied this to a wide range of geometric data structures in [8].

The above approaches share a common theme of leveraging the trust provided by a few digest values from a trusted party over multiple hashes to efficiently protect the integrity of the digested content. All of these methods rely on a single trusted owner to compute the digest. Our scheme avoids this limitation, but extending these results to a more complex distributed setting introduces new challenges to maintaining efficient data authentication. Nonetheless, our protocols incur essentially the same computational effort as in the non-distributed setting.

Two recent papers have started to reduce the need for a single trusted data owner. Buldas, Laud and Lipmaa [2] presented a scheme for accountable certificate management. Though the focus and setting are different from ours, they achieve some similar results. They show how an untrusted party can compute a digest of a set of certificates and, for a given certificate, prove that it is or is not a member of the set. Using an *authenticated search tree,* they show that unless an adversary finds a collision in the hash function, no contradictory proof can be given. This is almost the same as our result in the untrusted owner setting described in section 3 for selection queries. Our results, however, describe extensions to more complex queries and methods to prevent data which is falsely attributed to an owner. In [3], Buldas, Roos and Willemson provide some detail for a solution to "interval" or single dimensional range queries, but do not address the issue of proofs to owners showing that data is correctly represented. They describe as an open problem the extension of authentication techniques to multi-dimensional range trees. We solve this open problem using a *count certified search tree* which is an enhanced version of their authenticated search tree. Our mechanisms guarantee that the untrusted third party has computed the digest of a search tree just as a fully trusted party would. This is a stronger guarantee which allows a richer range of queries in our setting.

2. Implementations for Trusted Owners

Suppose a Publisher wants to collect data from n data owners, store this data in a Binary Search Tree (BST) and then compute a digest of the data. We present an efficient and secure protocol which allows the n owners to determine that the digest computed is the same as the digest which would be computed by a trusted party with access to the owners' joint data set. We extend this to digests of Multidimensional range trees (MDRTs) to efficiently answer multi-attribute queries.

We assume throughout the paper that all data items d have a value $\mathsf{key}(d)$ which is unique. It is possible that two data owners may each have data items with the same local key values. By adding a unique owner identifier to the local key values, the new global key value $\mathsf{key}(d)$ is again unique. We give a somewhat informal description of our results in this section to focus on the main ideas. A more formal treatment (along with the proofs) is given in the full version [16].

2.1. Owner Certification of a Binary Search Tree

We consider a setting with n owners with data sets $D_1 \dots D_n$. We let D denote this combined data set and let N be the total number of items. For the protocol to produce a correct result, the true value of N must be used in the Owner verification. We discuss ways to ensure that each Owner knows the correct value of N in section 2.4.

Following the protocol outlined in the introduction, our goal is to allow an untrusted Publisher who knows D to construct a binary search tree and its digest Σ, then efficiently prove to the owners (using the proofs π^i) that their data is correctly represented in a tree which has digest Σ. Once we establish that the digest Σ correctly represents D, the Publisher can answer queries from users and provide proofs just as in single owner Authentic Publication. If all parties follow the protocol correctly, then all owners will accept their proofs for Σ. The following theorem states that our *Owner BST certification protocol* is secure.

Theorem 1 *For a set of n owners with data sets $D_1 \ldots D_n$, where N is the number of data items in $D = \bigcup_{i=1}^{n} D_i$, if each Owner correctly executes the BST certification protocol and accepts his proof for the digest Σ and the data count N, then either Σ is a valid digest of a BST for D, or the Publisher has found a collision in the hash function used.*

The Publisher can compute the digest Σ and all proofs in $O(N \log N)$ time, and Owner i can verify his proof in $O(|D_i| \log N)$ time.

Thus the verification time for an owner i is nearly linear in the size of the set D_i. The full details of the proof of Theorem 1 are in the full paper [16]. However, the next two subsections, describe the digesting scheme and sketch how the publisher proves to the owners that their data is correctly represented.

2.2. The Count Certified Search Tree

Since each owner only sees a small part of the tree, we need a stronger digesting scheme than is used in a standard Merkle tree. A *count certified search tree* is a binary search tree, storing data values at the leaves and split values at internal nodes and is digested using the following scheme. The *count* associated with a node v of the tree (denoted $\mathsf{count}(v)$) is the number of data items stored in the leaves of the subtree rooted at v. We also let $\mathsf{split}(v)$ be the split value and $\mathsf{digest}(v)$ be the digest value. Our digesting scheme uses a collision-resistant hash function H. The digest value of a leaf is just the hash of its associated data value. For an internal node v with children u and w,

$$\mathsf{digest}(v) = \mathsf{H}(\mathsf{split}(v), \mathsf{count}(u), \mathsf{count}(w), \mathsf{digest}(u), \mathsf{digest}(w)).$$

We note that the BST structure is only a logical structure which the protocol must use for digesting and proofs. The publisher need not physically implement his database this way. For example, prior results show that a logical BST can be used with a physical B-tree implementation [11, 10, 12].

2.3. Proofs to Owners

We sketch how the Publisher proves to an owner O_i with data set D_i that this data is properly represented. Details are presented in [16]. For each data item d in D_i the Publisher's proof essentially simulates a search for data item d: the proof is a sequence of five tuples of values followed by d. The first five tuple contains the five values used to compute the digest value Σ of the root r of the (claimed) BST with left child u and right child w: $\mathsf{split}(r)$, $\mathsf{count}(u)$, $\mathsf{count}(w)$, $\mathsf{digest}(u)$, $\mathsf{digest}(w)$.

The owner hashes these five values to get a result x, and rejects unless $x = \Sigma$. If $x = \Sigma$, he uses the root's split value to determine if the search should go left, $d \leq \mathsf{split}(r)$, or right, $d > \mathsf{split}(r)$. If the search goes left, the next five values of a correct proof hash to $\mathsf{digest}(u)$.

This digest value is known from the first five values in the proof. Thus we can check that the second five values in the proof hash to $\mathsf{digest}(u)$, and we can also check that the counts match (the second five values should include the number of leaves in the left and right subtrees of u; the sum of these should match $\mathsf{count}(u)$, the number of leaves in the subtree at u, which was given in the first five values of the proof).

The verification process continues until, in the final step, we should reach a leaf whose value is d and which has an associated count of one.

In [2, 12] they also use split values to simulate a search however, our use of the counts is new. The split values make sure that no data values are missing. The counts make sure that no extra data items have been included. At each step of the proof the owner checks that the sizes of the next two subtrees along the path sum to the size of their parent. To make the whole process work, we also need to check that the size of the root (the sum of the counts of its two subtrees) matches *N*, the total size of the data set. We discuss these issues in more detail in the next section.

2.4. Implementation Issues and Assumptions

So far we have concentrated on the owner-publisher procedures and their proofs, but not the underlying communication mechanisms necessary to carry out the protocols. Theorem 1 shows that if all n owners follow the protocol correctly and approve their proofs for the digest value Σ and for the correct data count value *N*, then users can be confident that the digest Σ is for a BST which represents the combined data set. We now address the related issues of how to ensure that a User:

1 knows all Owners approve the digest Σ for the correct value of *N*.

2 gets the digest Σ.

This will partly depend on what we assume the user knows about the n owners and what the owners know about each other. At one extreme we can assume a user knows the identity of all participating owners along with the owners' public keys, and thus could expect to see a signed approval from each owner. At the other extreme, a user may know none of the participating owners, but may relay on a trusted third party "moderator" to determine an appropriate set of owners to participate. In between, a user may know of some owners they want involved, but expect other owners to contribute as well.

USING A TRUSTED MODERATOR. A trusted *Moderator* reduces the information both users and owners need about the set of owners. This is like an online Better Business organization which accepts owners who agree to follow the protocols. After an owner gets a correct proof for a digest value Σ, he sends a signed message to the Moderator: his approval, the digest, the size of his data set, the claimed total size of the entire data set, and the Publisher's web address. Once the Moderator gets an approval from all owners, it is easy to check that all owners approved the same digest and that the total of the individual sizes of the data sets matches the global data set size. The digest Σ for that Publisher can then be approved and posted (or, the Publisher gets a signed approval from the Moderator which can be given to users).

With this scheme a user can get answers and have confidence that the digest they are using is correct without knowing anything in advance about the set of owners. They only need to know about (and trust) the Moderator, who can then provide a signed approval of the digest Σ to the Publisher who passes this on to the Users. Thus, the Moderator has much less overhead than the Publisher since he does not directly deal with the Owners' data sets (just a single acknowledgment per owner) and also need not deal directly with Users.

ELIMINATING THE TRUSTED MODERATOR USING PKI. If we assume that users know which owners are involved and the owners' public keys, we can eliminate the need for a trusted Moderator. In this case, the publisher can give the user each owner's signed accept/reject response to Σ. The response would contain Σ and the count value each owner verified. The User would only need to know that the true count value was used to be confident that the Publisher's computed digest is accurate. Any number of assumptions could be made that ensure that the correct data count value N is used, but a simple and reasonable assumption is that at least one Owner knows N. The User simply checks that all Owners agree on the count value. Even if no Owner initially knows N, there are a number of efficient methods that a set of Owners could use to determine N. Any such method is sufficient to ensure the correctness of the Publisher computed digest value.

2.5. Multi-dimensional Range Trees

For a database containing a set of N r-tuples, an r-dimensional range query has the form $(l_1, u_1), \ldots, (l_r, u_r)$. The answer is the set of points with all coordinates satisfying the corresponding range $\{(x_1, \ldots, x_r) : l_i \leq x_i \leq u_i \text{ for } 1 \leq i \leq r\}$. These r-tuples can be efficiently found using a *multidimensional range tree* (MDRT) in time $O(\log^r N + k)$ where k is the number of r-tuples in the answer set [1]. This important setting allows us to support multi-attribute queries. In a 2D MDRT each data item d is stored in $\log N$ ordinary BSTs (all tied together in a larger data structure). Thus for each data item d an owner i gets a proof Π_d^i which has $\log N$ proofs similar to those used for a data item in the BST setting. The proof also has additional information tying the $\log N$ pieces together. Extending this result to higher dimensions, we get the following theorem:

Theorem 2 *For a set of n owners with data sets $D_1 \ldots D_n$, where N is the number of data items in $D = \bigcup_{i=1}^n D_i$, if each Owner correctly executes the MDRT certification protocol and accepts his proof for the digest Σ of an r-dimensional range tree, then either Σ is the digest of an r-dimensional range tree over D or the Publisher has found a collision in the hash function H.*

The Publisher can compute the digest Σ and all proofs in $O(N \log^r N)$ time, and Owner i can verify his proof in $O(|D_i| \log^r N)$ time.

3. Protocols for the Untrusted Owners Setting

In this setting there may be "untrusted owners" who may not follow the protocol. These owners may conspire with the publisher or other untrusted Owners to present false results.

An owner does not see the publisher's entire database, so cannot depend on the accuracy of data other than his own. However, an owner still wants assurance that any of his data that matches a query will be properly included in the answer, and that data not in his data set cannot be falsely attributed to him. We continue to assume that all data items have an identifier for the owner. Assuming no collisions in the hash function are found, we show how to achieve the following guarantee for an honest owner when a user submits a single or multi-dimensional range query: if an owner O_i accepts the publisher's proof for D_i and the user accepts the proof for the answer given to a query, then the data in the answer attributed to O_i is exactly what O_i should have returned if he had answered the query only on his data set D_i. Removing the assumption that all participating owners are trusted motivates our focus on individual owners in the relaxed security statement.[1] We can also extend this result to multi-dimensional range queries in a fairly straightforward way. We state the one-dimensional result formally in theorem 3.

Theorem 3 *Let $\mathsf{F}([a,b],D)$ be a function which returns answers to one dimensional range queries on the collection of data sets D. If an owner O_i correctly executes the Owner BST certification protocol and accepts the proof of the digest Σ and the user correctly accepts the answer returned by the publisher, denoted $\widetilde{\mathsf{ans}}$, then either 1) $\mathsf{F}([a,b],D_i) = \widetilde{\mathsf{ans}} \cap D_i$ or 2) the Publisher has found a collision in the hash function.*

3.1. Preventing Falsely Attributed Data

We frequently want to prevent a dishonest publisher from giving the user data which is attributed to an honest owner, but is not in the owner's data set. We can prevent such "false attribution" by adding an additional mechanism to our protocols.

We describe a variation of our results in the untrusted setting where each data element is checked efficiently using a single hash computation. In the new approach, each data owner securely combines each data element with a secret value to form a means to authenticate the origin of that data element, the owner commits to the secret, and later the owner reveals the secret so that users can use the secret to authenticate the origin of the data elements. More specifically the protocol is as follows.

1 Each owner, O_i ,generates a secret random number, k_i.[2]

2 O_i computes a message authentication code (MAC), $h_{k_i}(d)$, for each data element $d \in D_i$. The MAC is the hash of each data element d using a hash function keyed with the unrevealed random number k_i.[3]

3 O_i passes each of his data elements and its corresponding message authentication code to the publisher. Also, O_i passes a commit-

[1] In the presence of untrusted owners, and thus untrusted data, answers to queries which depend on the entire data set are no longer reliable. For example, the minimum of the data values is not useful since spurious data values could cause skewed results.

[2] The size of k_i should be chosen, with respect to the message authentication function in the next step, to be sufficiently large to prevent a variety of attacks on the system.

[3] Note that we assume that $\mathsf{key}(d)$ makes it easy to determine the claimed owner of the data element, so the user can efficiently identify the k_i of the owner. Otherwise, the user would have to check each k_i of all possible owners to determine the owner.

ment to k_i of the form $salt_i, h_{k_i}(salt_i)$. The publisher now builds the Count Certified BST using as leaf values the pairs $(d,\ h_{k_i}(d))$ which are still ordered by $\mathsf{key}(d)$.

4 After receiving the inputs from all owners, the publisher commits to the root hash Σ of the combined database, the list of data owners, and the commitments of each data owner, [4]
$\Sigma_{pub} = (\Sigma, O_1 \ldots O_j, [salt_1, h_{k_1}(salt_1)], \ldots, [salt_j, h_{k_j}(salt_j)])$.

5 O_i checks that the root hash properly reflects his data values essentially as in the prior section, and that $(salt_i, h_{k_i}(salt_i))$ is correct. He then reveals k_i to the publisher along with his signed acceptance of Σ_{pub}. The publisher can now make the tuples (O_i, k_i) available to the users in query responses along with O_i's signed acceptance of Σ_{pub}. To verify the authenticity of a data element which appears to belong to O_i, the user confirms that O_i signed Σ_{pub} and checks that k_i corresponds to the O_i's secret commitment.[5] Finally, the user verifies the message authentication code on the data element using the corresponding k_i.

It takes $O(1)$ effort to check $d \in D_i$ due to the hash commitment scheme and so for selection and range queries with T answers, verification now takes $O(\log N + T)$ time and space for a 1D range query and for r-dimensional range queries, verification takes $O(\log^r N + T)$.

4. Conclusion

One main limitation of our scheme is the need for users to know who has approved the digest. This is likely to make updates harder and introduces extra overhead for the user and owners. In addition, we would like to extend our results to a wider class of query types, as was done for the single owner case in [12].

Acknowledgments

The authors would like to thank Phil Rogaway for his excellent suggestions in the preliminary development of this work. Matt Franklin

[4]We might also want to include a time as part of Σ_{pub} as this would give the user evidence of the freshness of the values used in query answers. Even stronger would be to provide a time interval during which the values are presumed valid. Note that we typically assume that the data used in these applications does not change quickly, so precise clock synchronization is not required.
[5]Note, these steps need to be performed at most once per publication since notes corresponding to successful verification might be cached with the user.

and Prem Devanbu also provided valuable feedback for this paper. This work was supported by NSF grant CCR 85961.

References

[1] M. D. Berg, M. V. Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry.* Springer, New York, 2000.

[2] A. Buldas, P. Laud, and H Lipmaa. Eliminating counterevidence with applications to accountable certificate management. *Journal of Computer Security,* 10:273–296, 2002.

[3] A. Buldas, M. Roos, and J. Willemson. Undeniable replies for database queries. In *Proceedings of the Fifth International Baltic Conference on DB and IS,* volume 2, pages 215–226, 2002.

[4] S. Charanjit and M. Yung. Paytree: Amortized signature for flexible micropayments. *Second Usenix Workshop on Electronic Commerce Proceedings,* 1996.

[5] P. Devanbu, M. Gertz, A. Kwong, C. Martel, G. Nuckolls, and S. G. Stubblebine. Flexible authentication of xml documents. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS-8),* pages 136–145, 2001.

[6] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic publication over the internet. *Journal of Computer Security,* 3(11):291–314, 2003.

[7] P. Devanbu and S. Stubblebine. Stack and queue integrity on hostile platforms. *IEEE Transactions on Software Engineering,* 26(2), 2000.

[8] M. Goodrich, R. Tamassia, N. Triandopoulos, and R. Cohen. Authenticated data structures for graph and geometric searching. Technical report, Center for Geometric Computing, Brown University, 2002.

[9] S. Haber and W. S. Stornetta. How to timestamp a digital document. *J. of Cryptology,* 3(2), 1991.

[10] Petros Maniatis and Mary Baker. Enabling the archival storage of signed documents. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST 2002),* pages 31–45, Monterey, CA, USA, January 2002. USENIX Association.

[11] Petros Maniatis and Mary Baker. Secure history preservation through timeline entanglement. In *Proceedings of the 11th USENIX Security Symposium,* San Francisco, CA, USA, August 2002.

[12] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. Stubblebine. A general model for authentic data publication. To appear in *Algorithmica,* `http://truthsayer.cs.ucdavis.edu/pubs.html`.

[13] R.C. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy,* pages 122–134. IEEE Computer Society Press, 1980.

[14] R.C. Merkle. A certified digital signature. Advances in Cryptology–Crypto '89, *Lecture Notes in Computer Science,* 435:218–238, 1990.

[15] M. Naor and K. Nissim. Certificate revocation and certificate update. *Proceedings of the 7th USENIX Security Symposium,* 1998.

[16] G. Nuckolls, C. Martel, and S. Stubblebine. Certifying data from multiple sources. Available at `http://truthsayer.cs.ucdavis.edu/pubs.html`.