Volume: 2 Issue: XII Month of publication: December 2014
DOI:

Volume 2 Issue XII, December 2014
ISSN: 2321-9653

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Authentication of Data Structures for Graph

Anurag Sharma[#1], Vikram Singh[#2], Jaspal Yadav[#3]
Dronacharya College of Engineering

Abstract : Authenticated data structures provides cryptographic proofs where the answers are as accurate as the author intended, even if the data structure is maintained by a remote host. We have presented few techniques for authenticating data structures that represent graphs and geometric data's. A model has been given to represent the source and functions of working authentication of data structures for graph and geometric searching. We have also mention answering queries at distributed directories. When a user queries a directory, it receives a cryptographic proof in answer, where the proof contains statement by the source. We have also shown how to efficiently authenticate data for fundamental problems, such as on geometric objects, etc. this allows the design of new, efficient authenticated data structures for fundamental problem on network, such as path and connectivity queries over graphs.

I. INTRODUCTION

In this paper we are interested in studying a new dimension in data structure and algorithm checking. How can we design sophisticated data structures and algorithms so that their responses can be varied as accurately as if they were coming from their author, even when the response is coming from an un-trusted host? Digital signatures can be used to verify simple static documents, but are inefficient for dynamic data structures. We therefore need new techniques for authenticating data structures. In particular, we are interested in efficiently verifying paths and connectivity information in transport and computer networks (that is, combinatorial graph structures), even when the network is changing. In addition, we are interested in verifying complex geometric queries in spatial data bases, such as ray shooting queries, point location queries, and range searching queries, which are used extensively in geographic (1) Information systems. The main challenge in providing an integrity service in such contexts is that the space of possible answers is much larger than the data size itself. For example, there are $\boldsymbol{O}$ **(*n*2)** different paths in a tree of ***n*** nodes, and each of these paths can have $\boldsymbol{O}$ **(*n*) edges**. Requiring an authenticator to digitally sign every possible response is therefore prohibitive, especially when the data is changing due to the insertion or deletion of elements in the set. Ideally, we would like our authenticator to sign just a single digest of our data structure, with that digest being built from the careful combination of cryptographic hashes of subsets of our data. If we can achieve such a scheme, then verifying the answer to a query in our data base can be reduced to the problem of collecting the appropriate partial hashes for a user to recomputed the digest of the entire structure and compare that to the digest signed by the authenticator. Even when we follow this approach, however, we are faced with the challenge of how to subdivide the data in a way that allows for efficient assembly for any possible query. For simple data structures, such as dictionaries, this subdivision is fairly straightforward (say using a linear ordering and a Markel hash tree) but the subdivision method for complex structures, such as graphs, geometric structures, and structures built using the fractional cascading paradigm is far from obvious. For example, we have no linear ordering among the data in such cases upon which to build a hash tree.

II. MODEL OF AUTHENTICATION OF DATASTRUCTURE

The authenticated data structure model involves a structured collection **S** of objects (e.g., a set or a graph) and three parties: the source, the responder, and the user. A repertoire of query operations and optional update operations are assumed to be defined over **S**.

The role of each party is as follows (2):

A. The source holds the original version of **S**. Whenever an update is performed on **S**; the source produces structure authentication information, which consists of a signed time-stamped statement about the current version of **S**.

B. The responder maintains a copy of **S**. It interacts with the source by receiving from the source the updates performed on **S** together with the associated structure authentication information. The responder also interacts with the user by answering queries on **S** posed by the user. In addition to the answer to a query, the responder returns answer authentication information, which consists of **(i)** the latest structure authentication information issued by the source; and **(ii)** a proof of the answer.

C. The user poses queries on **S**, but instead of contacting the source directly, it contacts the responder. However, the user trusts the source and not the responder about **S**. Hence, it verifies the answer from the responder using the associated answer authentication information.

The data structures used by the source and the responder to store collection **S**, together with the algorithms for queries, updates, and verifications executed by the various parties, form what is called an authenticated data structure. In a practical deployment of an authenticated data structure, there would be several geographically distributed responders. Such a distribution scheme reduces latency, allows for load balancing, and reduces the risk of denial-of-service attacks. Scalability is achieved by increasing the number of responders, which do not require physical security since they are not trusted parties.

III. OVERVIEW ON MODEL

Throughout this section, we denote with n the size of the collection *S* maintained by an authenticated data structure.

Early work on authenticated data structures was motivated by the certificate revocation problem in public key infrastructure and focused on authenticated dictionaries, on which membership queries are performed.

The hash tree scheme introduced by Merkle (3) (4) can be used to implement a static authenticated dictionary. A hash tree ***T*** for a set ***S*** stores cryptographic hashes of the elements of ***S*** at the leaves of ***T*** and a value at each internal node, which is the result of computing a cryptographic hash function on the values of its children. The hash tree uses linear space and has ***O*(log *n*)** proof size, query time and verification time. A dynamic authenticated dictionary based on hash trees that achieves ***O*(log *n*)** update time is described in(5) dynamic authenticated dictionary that uses a hierarchical hashing technique over skip lists is presented in(6), this data structure also achieves ***O*(log *n*)** proof size, query time, update time and verification time. Other schemes based on variations of hash trees have been proposed in (2,6,7,8)

A detailed analysis of the efficiency of authenticated dictionary schemes based on hierarchical cryptographic hashing is conducted in(9,10) where precise measures of the computational overhead due to authentication are introduced.

IV. CRYPTO GRAPHICS

The basis of trust in our authentication model is the assumption that the user trusts the source. This is expressed by means of a digital signature scheme. Moreover, all the desired security results are achieved by means of the use of a cryptographic hash function. A cryptographic hash function **h** typically operates on a variable-length message **M** producing a fixed-length hash value h (**M**). We assume some well-defined binary representation for any data element **e**, so that **h** can operate one. Also, we assume that rules have been defined so that **h** can operate over any finite number of elements. A cryptographic hash function is a collision-resistant hash function, if, given the value **h(x)**, it is computationally intractable to find **x** and, moreover, if, given **y**, it is computationally intractable to find **x 6= y** with **h(x) = h(y)**. The collision resistance property will be used for our security results. Let **S** be a data set owned by the source. In our authentication schemes, a collision-resistant hash function is used to produce a digest, i.e., a cryptographic hash value over the data elements of **S**. The digest is computed in a systematic way which can be expressed by means of directed acyclic graph (DAG) defined over **S** (similar technique is presented in [11]. We define a single-sink DAG **G** associated with **S** as follows. Each node **u** of **G** stores a label **L(u)** such that if **u** is a source of **G**, then **L(u) = h(e1,...,em)**, where **e1,...,em** are elements of **S**, else (**u** is not a source of **G**) **L(u) = h(e1,...,en, L(z1),...,L(zk))**, where **(z1, u),...,(zk, u)** are edges of **G** and **e1,...,en** are elements of **S** (here, both **m** and **n** are some constant integers). We view the label **L(t)** of the sink **t** of **G** as the digest of **S**, which is computed via the above DAG **G**. We call the above scheme as hashing scheme of **S** using **G**. The answer to a query usually involves some elements. Then, the proof typically consists of the signed digest and all the information (labels of **G**) that is necessary for the re-computation, given the answer, of this digest by the user. The authentication techniques presented in this paper are based on the following general scheme. The source and the directory store identical copies of the data structure representing **S** and maintain the same hashing scheme on **S**. The source periodically signs the digest of **S** together with a timestamp and sends the signed time stamped digest to the directory. When the user poses a query, the directory returns to the user **(1)** the signed time stamped digest of **S**, **(2)** the answer to the query and **(3)** a proof consisting of a small collection of labels from the hashing scheme that allows the re-computation of the digest. The user validates the answer by re-computing the digest, checking that it is equal to the signed one and verifying the signature of the digest.

V. AUTHENTICATED BI-CONNECTIVITY QUERIES

Let **G** be a graph that is maintained through operations: make Vertex (***v***) and insert Edge (***u; v; e***) (add edge *e* between vertices ***u*** and ***v*** in **G**). We are interested in authenticating query operation are Bi-connected (**u; v**) that determines whether ***u*** and ***v*** are in the same bi-connected component. We extend the data structure of [11]. We maintain the block-cut vertex forest **B** of ***G***. Each tree ***T*** in ***B*** corresponds to a connected component of ***G***. There are two types of nodes in ***T*:** block nodes that correspond to blocks (bi-connected components) of ***G*** and vertex nodes that correspond to vertices of ***G.*** Each edge of ***T*** connects a vertex node to a block node. The block node associated with a block ***B*** is adjacent to the vertex nodes associated with the vertices of ***B***. We have that two vertices ***u*** and ***v*** of ***G*** are in the same bi-connected component if and only if there is a path between the vertex nodes of ***B*** associated with ***u*** and ***v*** and this path has length two. Thus, operation are Bi-connected in ***G*** is reduced to operation path Length in ***B.***

Theorem 1: Given a graph G with n vertices, there exists a dynamic authenticated data structure for bi-connectivity queries on G with the following performance: (i) query operation are Bi-connected takes O(log n) time, update operation make Vertex takes O(1) time and update operation insert Edge takes O(log n) amortized time; (ii) the query authentication information has size O(log n); (iii) the update authentication information has size O(1); and (iv) the query verification time is O(log n).

VI. AUTHENTICATED TRI-CONNECTIVITY QUERIES

We now show how to authenticate operation are Tri-connected (***u; v***) that reports whether there is a tri-connected component containing vertices ***u*** and ***v.*** We extend the data structure of [12], where a bi-connected graph (or component) ***G*** is associated with an ***SPQR*** *tree* ***T*** that represents a recursive decomposition of ***G*** by means of separation pairs of vertices. Each **S-, P-,** and **R-**node of ***T*** is associated with a tri-connected component ***C*** of ***G*** and stores a separation pair (***s; t***), where vertices **s** and **t** are called the poles of ***C***. A **Q-**node of ***T*** is associated with an edge of ***G***. Each vertex ***v*** of ***G*** is allocated at several nodes of ***T*** and has a unique proper allocation node in ***T.*** Our authenticated data structure augments tree ***T*** with **V-**nodes associated with the vertices of ***G*** and connects the **V-**node of a vertex ***v*** to the proper allocation node of ***v*** in ***T.*** Also, it uses node attributes to store the type (**S, P, Q, R, or** V) of a node of ***T*** and its poles. We can show that operation are Tri-connected can be reduced to a small number of path Length and includes Node Type queries on the augmented **SPQR** tree.

Theorem 2: Given a graph G with n nodes, there exists a dynamic authenticated data structure that supports tri-connectivity queries with the following performance: (i) query operation are Tri-connected takes O(log n) time, update operation make Vertex takes O(log n) time and update operation insert Edge takes O(log n) amortized time; (ii) the query authentication information has size O(log n); (iii) the update authentication information has size O(1); and (iv) the query verification time is O(log n).

VII.APPLICATIONS

In all of these problems, the underlying catalog graph has degree bounded by a small constant. In the following, *n* denotes the problem size.

Theorem 12: There is an authenticated data structure for answering line intersection queries on a polygon that can be constructed in O(n log n) time and uses O(n log n) storage. Denoting with k the output size,

queries are answered in O(log n + k) time; the answer authentication information has size O((k + 1) log $\frac{n}{k+1}$);

and the answer verification time is O((k + 1) log $\frac{n}{k+1}$).

Theorem 13: There are authenticated data structures for answering ray shooting and point location queries that can be constructed in O(log n) time and use O(n log n) storage. Queries are answered in O(log n) time; the answer authentication information has size O(log n); and the answer verification time is O(log n).

All these results have applications to the authentication of geographic information systems.

REFERENCES

[1] Research supported in part by DARPA Grant F30602 {00{2{0509.*y*University of California, Irvine, goodrich@acm.org *z* Brown University, *f*rt,nikosg@cs.brown.edu

*X*Algo Magic Technologies, rfc@algomagic.com.

[2] G. Di. Battista and U. Z. wick (Eds.): ESA 2003, LNCS 2832, pp. 2–5, 2003.cSpringer-Verlag Berlin Heidelberg 2003.

[3] R. C. Merkle Protocols for public key cryptosystems. In Proc Symp on Security and Privacy pages 122–134. IEEE Computer Society Press, 1980.

[4] R. C. Merkle A certified digital signature. In G. Brassard, editor, Proc. CRYPTO '89, volume 435 of LNCS, pages 218–238. Springer Over lag, 1990.

[5] M Naor and K. Nissim. Certificate revocation and certificate update. In *Proc. 7th USENIX Security Symposium*, pages 217–228, Berkeley, 1998.).

[6] T Good rich and R. Tamassia. Efficient authenticated dictionaries with skip lists and commutative hashing. Technical reports Johns Hopkins Information Security Institute, 2000. http://www.cs.brown.edu/cgc/stms/papers/hashskip.pdf.)

[7] A Bullas, P. Laud, and H. Limas. Accountable certificates management using undeniable attestations. In *ACM Conference on Computer and Communications Security*, pages 9–18. ACM Press, 2000. 6.I. Gassko, P. S. Gemmell, and P. MacKenzie. Efficient and fresh certification. *Int. Workshop on Practice and Theory in Public Key Cryptography (PKC '2000)*, volume 1751 of *LNCS*, pages 342–353. Springer Verlag, 2000.

8. P. C. Kocher. On certificate revocation and validation. In *Proc. Int. Conf. on Financial Cryptography*, volume 1465 of *LNCS*. Springer-Verlag, 1998.)

9. Tamassia and N. Triandopoulos. On the cost of authenticated data structures. Technical report, Center for Geometric Computing, Brown University, 2003. http://www.cs.brown.edu/cgc/stms/papers/costauth.pdf.)

10. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. Stubblebine. A general model for authentic data publication, 2001. http://www.cs.ucdavis.edu/~devanbu/files/model-paper.pdf.

[11.] J. Westbrook and R. E. Tarjan. Maintaining bridge-connected and biconnected components on-line. *Algorithmica*.

[12.] G.Di Battista and R. Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*.