Rolling Boles, Optimal XML Structure Integrity for Updating Operations

Sebastian Graf
University of Konstanz
Departement of Computer and Information Science
Konstanz, Germany
sebastian.graf@uni-konstanz.de

Sebastian Kay Belle
University of Konstanz
Departement of Computer and Information Science
Konstanz, Germany
sebastian.kay.belle@uni-konstanz.de

Marcel Waldvogel
University of Konstanz
Departement of Computer and Information Science
Konstanz, Germany
marcel.waldvogel@uni-konstanz.de

ABSTRACT

While multiple techniques exist to utilize the tree structure of the *Extensible Markup Language*(XML) regarding integrity checks, they all rely on adaptions of the *Merkle Tree*: All children are acting as one slice regarding the checksum of one node with the help of an one-way hash concatenation. This results in postorder traversals regarding the (re-)computation of the integrity structure within modification operations. With our approach we perform nearly in-time updates of the entire integrity structure. We therefore equipped an *XHash*-based approach with an incremental hash function. This replaces postorder traversals by adapting only the incremental modifications to the checksums of a node and its ancestors. With experimental results we prove that our approach only generates a constant overhead depending on the depth of the tree while native *DOMHash* implementations produce an overhead based on the depth and the number of all nodes in the tree. Consequently, our approach called *Rolling Boles* generates sustainable impact since it facilitates instant integrity updates in constant time.

Categories and Subject Descriptors

H.2 [**Database Management**]: General—*Security, Integrity and Protection*

Keywords

XML, XML data integrity, DOMHash, XHash

General Terms

Design, Performance, Security, Verification

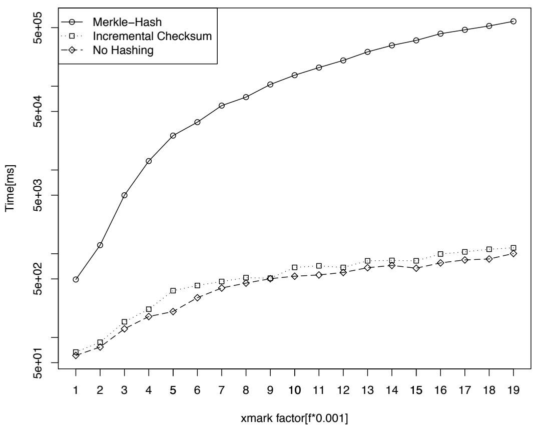
1. INTRODUCTION

The *eXtensible Markup Language* offers, due to its natural structure, great possibilities to ensure data integrity in constant time. Regarding subtree comparisons, security aspects and evaluation of changes between modifications, XML offers great benefits when equipped with an integrity structure relying on the well-know hash structure from Merkle [2]. Applied as a uniform independent interface language to combine heterogeneous technologies, integrity guarded XML has the ability to play a mandatory role in the universe of the *WWW*. This beneficial feature comes at a price: Each operation on the content of a node as well as on the structure of the entire tree must result in an adaption of the recursive hash tree in order to keep the integrity functionality. *Merkle Tree* hashes (as well as possible extension based on XML) always rely on the checksums of all children as well as on its own content. Each update operation needs a postorder traversal on the tree to adapt the checksums of the edited node plus its ancestors. Therefore, the price of integrity is a deceleration of the update operations in relation to the size of the structure. To minimize the necessary operations we boil down the update operations to the absolute necessary operation, namely the writing of the ancestors. By replacing the read operation of the siblings with a concatenation function for computing the hash of one node, we ensure an update performance of the integrity structure related to the path to the root and not to the size of the structure. We call this approach *Rolling Boles*. We show that our approach scales with a constant time overhead when we parse *XMark* instances while a native *Merkle Tree* implementation is generating an overhead related to the overall size of the instance. While normal approaches can secure the entire structure against any modification and access, our approach suits best when it comes to use cases where the integrity of an isomorphic subtree can result in equality plus when it comes to high update loads of the content or the structure.

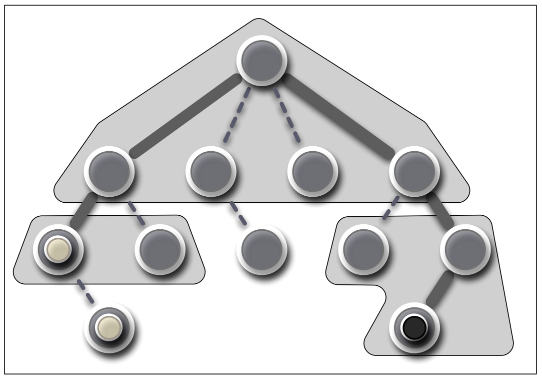
2. RELATED WORK

Data integrity related to XML as a tree structure become quick practicable with the *DOMHash* [1] where a simple DOM structure is checked by the recursive integrity of all element nodes. XML-equipped checksumming became common when it comes to finding differences between two versions of the same XML. When working with recursive checksums, some approaches [4] only care about isomorphic structures which are defined as equality regarding their checksums and their children which can differ in their relative position to each other. *Rolling Boles* relies on this approach by utilizing *XHash* which builds up a recursive, order independent structure of the single checksums.

Copyright is held by the author/owner(s).
WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0637-9/11/03.



(a) Incremental Insert of different XMark instances



(b) Schema of Insert/Remove

Figure 1: Benchmark and Schema Results

3. ROLLING BOLES

Rolling Boles works in a straight-forward way by utilizing one-way hash functions with two-way checksum concatenation. The checksum of each node is computed based on an hashing operation of the node content. The resulting checksum of the node is afterwards concatenated with the checksums of its children. In contrast of existing techniques, this concatenation is based on a two-way operation. That means that each checksum of a child is associated with the checksum of a node in an incremental way. This enables *Rolling Boles* to act in an optimized way with respect to any update operation of either the structure or the content due to the fact that operations on the children result in only read and write operations of the designated node.

Figure 1b shows examples of an insert operation as well as of a node removal. In the left side of the schema, a new subtree, denoted by white nodes, is inserted. Due to the incremental adaptivity of *Rolling Boles*, only the nodes on the path to the root, which is marked by a thick line, have to be read and written. In opposite, the left and central grey areas denote the nodes which have to be read regarding normal postorder based approaches. When it comes to removal operations, exemplary shown in the right side of the figure, the scaling is similar. The black node is removed from the overall tree structure. Due to our incremental architecture, we only have to adapt the nodes to the path of root while a normal approach needs to adapt all nodes in the right and central area.

Even if a two-way concatenation of checksums represents a weakness on our approach, we believe that for simple, non-cryptographically checksum approaches, *Rolling Boles* represents a tool for instant checksum even with respect to heavy update operations.

4. RESULTS

Figure 1a shows the insertion time of different *XMark* [3] instances with increasing sizes. The integrity structure is maintained directly within the insertion of every single node. It is clearly visible that current approaches which rely directly on the checksums of the children generate a linear overhead related to the instance size. Since these approaches have to scan all children of one node in order to regenerate a checksum, they all rely on the number of all nodes in a tree. Therefore they are not able to generate only a constant overhead like *Rolling Boles* where the effort to update the checksums is only based on the depth of the node inserted.

5. FUTURE WORK AND CONCLUSION

Structural integrity is one major issue in *WWW* environments especially in the context of continuous modifications of resources. XML equipped with *Rolling Boles* is able to fulfill this task with only constant additional effort. This effort is independent of the number of nodes in the tree which enables *Rolling Boles* to maintain the integrity structure within single modification operations. Especially regarding stateless or atomic modifications, *Rolling Boles* decreases the overhead which goes along with a checksummed structure while current approaches toils with instant consistency of the integrity structure since the recursive definition of the *Merkle Hash* plus their adaption to XML are in need of all children when maintaining the checksum of a single node. This behaviour bounds current approaches to the overall number of nodes while *Rolling Boles* only relies on the depth of the node modified.

6. REFERENCES

[1] H. Maruyama, K. Tamura, and N. Uramoto. Digest values for DOM (DOMHASH). RFC 2803, Internet Engineering Task Force, Apr. 2000.

[2] R. C. Merkle. A digital signature based on a conventional encryption function. In *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, 1987.

[3] A. Schmidt, F. Waas, M. Kersten, M. J. Carey, I. Manolescu, and R. Busse. Xmark: A benchmark for xml data management. In *International Conference on Very Large Data Bases*, 2002.

[4] Y. Wang, D. DeWitt, and J. Cai. X-Diff: An effective change detection algorithm for XML documents. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 519–530. IEEE, 2004.