COMPUTERS & SECURITY 53 (2015) 203–214

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

New models for efficient authenticated dictionaries

Kevin Atighehchi, Alexis Bonnecaze, Gabriel Risterucci*

Aix Marseille University, CNRS, Centrale Marseille, I2M, UMR 7373, 13453 Marseille, France

ARTICLE INFO

Article history:
Received 20 December 2014
Received in revised form
24 March 2015
Accepted 23 April 2015
Available online 5 May 2015

Keywords:
Authenticated dictionary
Data structure
Merkle tree
Huffman code
Zipf
Time series

ABSTRACT

We propose models for data authentication which take into account the behavior of the clients who perform queries. Our models reduce the size of the authenticated proof when the frequency of the query corresponding to a given data is higher. Existing models implicitly assume the frequency distribution of queries to be uniform, but in reality, this distribution generally follows Zipf's law. Our models better reflect reality and the communication cost between clients and the server provider is reduced allowing the server to save bandwidth. The obtained gain on the average proof size compared to existing schemes depends on the parameter of Zipf law. The greater the parameter, the greater the gain. When the frequency distribution follows a perfect Zipf's law, we obtain a gain that can reach 26%. Experiments show the existence of applications for which Zipf parameter is greater than 1, leading to even higher gains.

1. Introduction

Authenticated dictionaries are used to organize and manage a collection of data in order to answer queries on these data and to certify the answers. They have been heavily studied recently and have many applications including certificate revocation in public key infrastructure (Naor and Nissim, 2000; Gassko et al., 2000), Web mail search results (Ohrimenko et al., 2012), geographic information system querying, or third party data publication on the Internet (Devanbu et al., 2001; Bayardo and Sorensen, 2005). This last application is of great interest with the advent of cloud computing and Web services. For example, it is important that a user who consults a Web page can be confident of the authenticity of that page (or some of its contents).

Classical schemes involve three actors (Tamassia and Triandopoulos, 2003; Goodrich et al., 2001): a trusted *source* which is generally the owner of the data, an untrusted provider also called *directory* and a set of *users* (also called clients). The directory receives a set of data from the source together with authentication information. These contents are stored by both the source and the directory but only the latter communicates with users. Therefore, as shown in Fig. 1, users communicate directly with the directory to query the authentication information on a given data. This information contains a cryptographic proof and allows the users to authenticate the data. Note that the source and the directory are not necessarily hosted on two distinct machines.

Most of authenticated dictionaries use Merkle trees, red–black trees or skip-lists as data structures. These

* *Corresponding author.*
E-mail addresses: kevin.atighehchi@univ-amu.fr (K. Atighehchi), alexis.bonnecaze@univ-amu.fr (A. Bonnecaze), gabriel.risterucci@gmail.com (G. Risterucci).
http://dx.doi.org/10.1016/j.cose.2015.04.010
0167-4048/

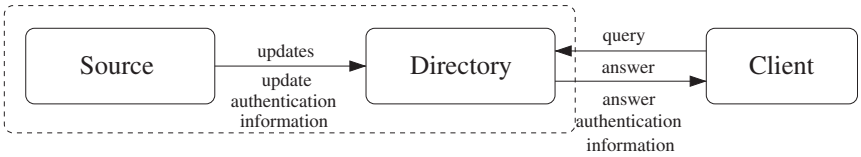

Fig. 1 – The three-party authentication model.

structures are closely equivalent in terms of cost of storage, communication and time (Tamassia and Triandopoulos, 2003). They are well adapted as long as no distinction is made between data. However, in some situations, it may be useful to manage data as a function of some parameters. In the case of publications on the Internet, some pages are accessed more frequently, depending on user behavior. Some pages have a better reputation than others, and it may prove useful to order them following this criterion. In fact, any behavioral criterion could be taken into account.

In this paper, we introduce authenticated dictionary schemes which take into account the frequency of data being accessed. As regards Web traffic, it is well known that its frequency distribution follows Zipf's law (Adamic and Huberman, 2002; Mahanti et al., 2013). More precisely, most traffic follows this law except for the traffic residue corresponding to very low frequencies. In fact, there is a drooping tail, which means that for these frequencies, the distribution decreases much faster than Zipf's law. Zipf law depends on a parameter (also called exponent). When this parameter is equal to 1, Zipf law is so-called perfect and the greater the parameter, the greater the gain.

The paper is organized as follows. Section 2 contains background information regarding data structures and the dictionary problem. Section 3 briefly presents the main types of existing dictionaries. In Section 4, we introduce the notion of frequency and we show its importance to optimize the efficiency of the dictionary, in particular in terms of authentication proof size. Since the frequency distribution varies over time, the way of updating the content of the data structure represents a major issue. A possible solution is to regularly reconstruct the structure. Another solution is to make some updates without reconstructing the whole structure. In the case of a reconstruction, a (current) frequency distribution must be considered since the structure depends on it. This frequency distribution can be calculated based on the preceding frequencies, either directly or using a predicting function. Section 5 focuses on the data structures that have to be used in order to take into account the frequency parameter. In Section 6, we propose the construction of a dictionary based on append/disjoin-only data structures. In Section 7, we introduce Huffman trees to improve efficiency. When the frequency distribution follows a perfect Zipf's law, we obtain a gain that can reach 26% on the average proof size compared to existing schemes. The use of an adaptive Huffman tree is discussed in Section 8. The decision of completely rebuild or just update the structure depends on the nature of the dictionary and its desired efficiency properties. In Section 9, we consider two use cases and we analyze the obtained gains. The first experiment gives non significant gains since Zipf law parameter is less than 1, whereas the second experiment exceeds the expected gains of a perfect Zipf law because the parameter is equal to 1.4.

2. Authenticated dictionary problem

2.1. Dictionary features and efficiency

The authenticated dictionary problem has already been defined in the literature, for example in Tamassia and Triandopoulos (2003), Goodrich and Tamassia (2001). In this section, we summarize the main features of an authenticated dictionary. The source has a set S of elements which evolves over time through insertion and deletion of items. The directory maintains a copy of this set and its role is to answer queries from the users. A user may request a given element or may perform a membership query on S in order to know whether an item belongs or not to S. The user must be able to verify the attached cryptographic proof (in particular, public information about the source must be available).

Efficiency makes the difference between a good dictionary and a bad one. This efficiency can be measured in terms of computation cost, which is the time taken by the computation together with the cost of the hardware (memory space and bandwidth) used by the entities. The size of the proofs is perhaps the most important parameter since it plays a significant role on the interface bandwidth of the directory. Moreover, it may reduce the time for a user to verify the answer to a query. The time spent by the directory to answer a query is also an important parameter when the number of users is very large. Space used by the data structure as well as source to directory communication should be optimized. Finally, the time to perform an update should also be optimized.

In the rest of the article, n represents the number of data elements ($n:=\#S$) and H represents a cryptographic hash function. The unique identifier of an element is denoted Id_i for $i\in\{1,\ldots,n\}$, its hashed identifier is $u_j = H(Id_i)$ for $j\in\{1,\ldots,n\}$ such that the values u_j are ordered ascendingly. Its content is denoted C_j and its hashed value c_j.

2.2. Data structures and authentication

Data structures represent a way of storing and organizing data so that searching, adding or deleting operations can be done efficiently. A static structure has a size that cannot be changed and therefore it is not possible to delete or add any data a posteriori. However, the size of dynamic data structures can change allowing insertion and deletion operations. In this