



## A Survey on Authenticated Data Structure and Various Implementation methods

Vaishali Ingale<sup>1</sup>, Mohnish Chaudhary<sup>2</sup>, Rashmi Sharma<sup>3</sup>, Ravinder Kumar<sup>4</sup>, Vaishnavi Pimpalnerkar<sup>5</sup>

<sup>1</sup>Professor, Information Technology, Army Institute of Technology, Pune, Maharashtra, INDIA.

<sup>2,3,4,5</sup>Student, Information Technology, Army Institute of Technology, Pune, Maharashtra, INDIA.

**Abstract:** Authenticated Data Structure (ADS) is a data structure which provides an computational proof of a document to the client. ADS uses the cryptography along with the data structure to ensure the client gets an legitimate information from source even if untrusted party is involved. The client submits an online queries to a some third party, this untrusted data holder reply on behalf of data owner or source. There is a need of mechanism which can provide a proof to the client that this incoming document or data is a valid and not corrupted by hackers. This untrusted party validate reply by providing a solid proof using various method like hashing, encryption.

This survey paper provides you a different techniques to design authentic data with a methods like Skip list, Commutative Hashing, Merkle Tree, Streaming ADS etc.

**Keywords:** Authenticated data structures, Computational proof, digest, RSA, Hashing Algorithm, SHA.1

### I. Introduction

Data replication or data tampering is one of the reason why storing a data on remote servers or on cloud can cause change in data when it is received. To avoid replication and how a client can verify that the data received is the original data and not a replicated one?

Suppose stock prices from Mumbai Stock Exchange are received by the brokers or portals from where the customers receives stocks prices. A person or investor whose money is invested in stocks needs to have some kind of assurance that the stock prices received is identical to the prices of Mumbai Stock Exchange.



Figure 1.1(a) Three party model for ADS

It's a typical setting where three party entites are involved in a authenticated data structure, usually referred to as three-party model. A trusted party called source, owns a data structure, that is send along with some cryptographic proof to one or more untrusted parties, called servers. Clients issue data structure queries to the servers and wish to publicly verify the answers received from the servers, based only on the trust they have in the source. This trust is conveyed using a time-stamped signature on the digest of the data structure (a collision resistant succinct representation of the data structure, e.g., the root hash of a Merkle tree), that is made available by the source. During an update, the source needs just to compute the new digest, whereas the server needs to update the authenticated data structure as a whole.



Figure 1.2(b) Three party model for ADS

As Micheal T. Goodrich, Roberto Tamassia and Nikos [5] suggest Authenticated Data Structure needs to achieve following key points described in [8] [5] are: (a) *minimum cost of computation*: computation done at

each side of the entity(source, third party, client) should not be tedious and slow, memory required for computation should be small; (b) *less communication overhead* : communication between source to third party and third party to client should be concise and less time consuming; (c) *security*: computational proof should be provided to the client for verification.

## II. Previous and Related work in ADS

### • Certificate Revocation

The method of certificate revocation involves source and client, source sends signed certificate with a timestamp and hash of all the certificate revocation list (CRL) [8] to the client. In this case third party or untrusted person would be a client and then periodically it is send to untrusted party. A untrusted party then forwards the whole to user who requests for it. But this scheme is inefficient as sending the updated CRL and the answer authentication information size  $O(n)$ .

Micali[9] proposed an alternate approach where the source periodically sends list of all issued certificates with the signed time-stamped value of one way hash function. This approach is much more efficient than the earlier one and this approach helped to reduce the size of authentication to  $O(1)$  containing only certificate identifier value and hash value indicating its status.

### • Commutative Hashing

Micheal T. Goodrich, Roberto Tamassia[10] introduced commutative hashing. If we consider a cryptographic hashing function with two argument  $a$  and  $b$  such that  $h(a, b)$  is equal to an integer fixed value  $k$  which is less than  $a$  and  $b$ . A  $h(a, b)$  is a digest of a pair  $(a, b)$ . Similarly we can use  $h(a, b, c)$  to digest a triple  $(a, b, c)$  and so on. To ease the process of verification Goodrich and Tamassia introduced the concept of commutative hashing function[10]. The hash function is commutative if it follows the following condition

$$h(a, b) = h(b, a) \text{ for all } a \text{ and } b.$$

For example, the commutative hash function  $h'$  for a given collision resistant hashing function[10]  $f$  can be defined as

$$h(a, b) = f(\min\{a, b\}, \max\{a, b\})$$

### • Skip List

Goodrich and Tamassia[10] designed a data structure[10] to store a set of objects  $S$  efficiently and in specific order. It supports the standard operations[8] **find** (to get or retrieve an element), **insert** (to enter or update an element) and **delete** (to remove an element). The elements are stored in a succession of link list where initial list  $S_0$  contains all element ordered in respect to the key and each element is propagated to next list probabilistically. Tower is a collection of all entries for a given single key and the topmost entry of tower is called as plateau node[8]. The remaining nodes other than plateau node are called as tower nodes. By default each entry of a list is filled with  $-\infty$  and  $\infty$  (sentinels) and top sentinel is referred as root.

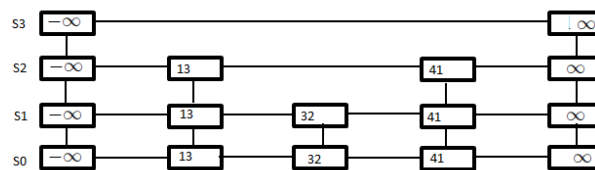


Figure 2.1 Example of Skip Lists

## III. Ways of Implementing ADS

### • Streaming Authenticated Data Structure

Charalampos Papamanthou, Elaine Shi, Roberto Tamassia, and Ke Yi[3] introduced the concept of streaming authenticated data structure. It is an expressive and efficient way and doesn't require direct communication between client and untrusted party(server) when the stream is observed. For Example, streaming  $x_1, x_2, x_3, x_4, \dots, x_n$  are stored in data structure which is held by the source and the client verifies and store only some local part which is needed of size  $O(\log n)$ [3]. When client sends a query to, subsequently server can provide the computation result and proof at  $O(\log M \log n)$

Where  $M$  is the size of ordered list from the elements are drawn and  $n$  is size of stream. The following properties are achieved using this ADS[3] :

(a) *Independence of client and server:*

When the  $x_1, x_2, x_3, x_4, \dots, x_n$  are streaming, client and server works independently with no interaction, the only interaction occurs only in query phase.

- (b) *Efficiency*: The running time of client and server during both update and queries as well as proof size is in term of log of M and n.
- (c) *Expressiveness*: Supporting wide range of queries over an ordered universe such as successor, range search and non-membership.

#### • Optimal Authenticated Data Structures with Multilinear Forms

Authenticated dictionary is defined as ‘*optimal*’ authenticated data structure because it does not add any additional asymptotic overhead to the corresponding plain non-authenticated DS. This authenticated dictionary is based on the new authentication primitive ‘*multi-linear forms*’ [1] proposed by Silverberg and Boneh [12].

The multi-linear form cryptographic primitive is described as a multi version of bilinear map. Bilinear map tool is used for solving the Diffie-Hellman problem, eventually it was described as a useful tool for cryptography [1]. The main drawback of bilinear form was that it cannot be fed back into map, finding self-bilinear map which can be fed back to a map used to construct multi-linear forms.

### IV. Merkle Tree And Authentication using RSA

#### • Merkle Tree

Merkle tree or Hash tree is a tree in which every non-leaf node consists of a digest which is hash of the concatenation of digest of its children node and leaf node is paired with digest of associated element. Merkle trees are useful in ADS because they provide a correct and secure content verification of data structure. If the height of merkle tree is known [6], the merkle tree can be represented in array using formula

$$M = 2^{h-1}$$

Where  $M = a_1, a_2, a_3, a_4, \dots, a_n$ ;

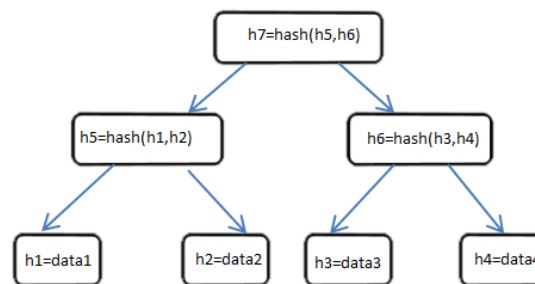


Figure 4.1 Example of Merkle Tree

Considering above example, Merkle tree of height equals to 3 is given. Each leaf node is associated with data value data1, data2, ..., and each non-leaf node contains digest of concatenation of hash of its children [6]. Like here h5 is the digest h1 and h2, h1 and h2 are digest of its left and right child respectively. When a query is sent to retrieve a value from tree at a position ‘n’, server retrieves a value from a position ‘n’ and along with result it sends a set of digest which is needed to compute root. The verifier at the client receives proof and result. The verifier keeps the copy of the proof and recomputes the digest to compare it with received proof.

#### A. Hashing Algorithm

When data is present at a remote area there is a chance that data may be lost or modified and third-party computations may not be performed correctly due to errors, opportunistic behaviour, or malicious attacks.

| Algorithm and Different Properties | Output Size | Input Size  | Rounds | Block Size |
|------------------------------------|-------------|-------------|--------|------------|
| MD5                                | 128         | $2^{64}-1$  | 64     | 512        |
| SHA0                               | 160         | $2^{64}-1$  | 80     | 512        |
| SHA1                               | 160         | $2^{64}-1$  | 80     | 512        |
| SHA2                               |             |             |        |            |
| SHA 224                            | 224         | $2^{64}-1$  | 64     | 512        |
| SHA 256                            | 256         | $2^{64}-1$  | 64     | 512        |
| SHA 384                            | 384         | $2^{64}-1$  | 80     | 1024       |
| SHA 512                            | 512         | $2^{128}-1$ | 80     | 1024       |
| SHA 512/224                        | 224         | $2^{128}-1$ | 80     | 1024       |
| SHA 512/256                        | 256         | $2^{128}-1$ | 80     | 1024       |
| SHA2                               |             |             |        |            |
| SHA 224                            | 224         | $\infty$    | 24     | 1152       |
| SHA 256                            | 256         |             |        | 1088       |
| SHA 384                            | 384         |             |        | 832        |
| SHA 512                            | 512         |             |        | 576        |

Figure 4.2 Comparison between Hash Algorithms

Thus storing data at remote area is attractive, client needs integrity guarantee in order to fully adopt this new paradigm and to maintain data integrity *Hashing Algorithms* like MD5, SHA0, SHA1, SHA2, SHA3 are used. Hashing Algorithm are used to generate digital data of fixed size from arbitrary size data. The output and input size of each algorithm differs like MD5 generates a digest of 128bit and input document could be anything. Similarly SHA1 takes document of size  $< 2^{64}$  and generates a output digest of 160bit.

#### • **RSA Algorithm**

Rivest-Shamir-Adleman (RSA) is a public-key encryption algorithm, where two separate keys are used for both encryption and decryption. RSA algorithm is widely used algorithm for a secure communication over insecure network. In RSA cryptography, when public key is used encryption the opposite key i.e private key is used for decryption and vice versa. There are variety of cryptography algorithm other than RSA like Rabin Cryptosystem, ElGMAI, NITRUEncrypt, Diffie-Hellman, McEliece. The advantage of using RSA is over other algorithm is that it follows a standard (PKCS#1) which states where every byte should go plus RSA private key is not easy to break because of the large integer number. For performance, RSA public key operation is very fast than other public key operation. This alone is can explain the wide acceptance this algorithm in industry.

### **V. Conclusion**

The advancement in ADS is changing the horizon of security over the network and ultimately resulting in maintain the integrity of data. The used algorithm likes SHA1 and RSA are helpful for a secure transmission of data over the network like internet. There are tremendous opportunities to work here in this arena and to make contribution.

In this paper we have concluded that among all the methods that we surveyed to implement ADS, implementation using Merkle Tree is more efficient because size of proof always remain  $\log_2 n$  and computational cost required by client and server is  $\log_2 n$ . An overview of Authenticated Data Structure and how it can be used for security purposes when data is stored at remote area with an untrusted party. We gone through the various method of implementation of ADS and concluded our study.

### **VI. Reference**

- [1]. Charalampos Papamanthou, Roberto Tamassia<sup>1</sup>, and Triandopoulos : Optimal Authenticated Data Structures with Multilinear Forms: Department of Computer Science, Brown University, Providence RI, USA RSA Laboratories, Cambridge MA, USA.
- [2]. Dan Rosenberg: On-Disk Authenticated Data Structures for Verifying Data Integrity on Outsourced File Storage.
- [3]. Charalampos Papamanthou, Elaine Shi, Roberto Tamassia, and Ke Yi: Streaming Authenticated Data Structures.
- [4]. Roberto Tamassia: Authenticated Data Structures: Department of Computer Science: Brown University Providence, RI 02912–1910, USA
- [5]. Michael T. Goodrichy Roberto Tamassiaz Nikos Triandopoulosz Robert Cohe: Authenticated Data Structures for Graph and Geometric Searching: University of California, Irvine, Brown University, Boston University. August 28, 2009.
- [6]. Andrew Miller, Michael Hicks, Jonathan Katz, and Elaine Shi: Authenticated Data Structures, Generically: University of Maryland, College Park, USA.
- [7]. Charalampos Papamanthou, Ph.D.,: Cryptography for Efficiency: New Directions in Authenticated Data Structures: Brown University, May 2011.
- [8]. Michael T. Goodrich, Roberto Tamassia, Andrew Schwerin: Implementation of an Authenticated Dictionary with Skip Lists and Commutative Hashing: Department of Computer Science Johns Hopkins University Baltimore, Maryland 21218, Department of Computer Science Brown University Providence, Rhode Island 02912.
- [9]. S. Micali. Efficient certificate revocation. Technical Report TM-542b, MIT Laboratory for Computer Science, 1996.
- [10]. M. T. Goodrich and R. Tamassia: Efficient authenticated dictionaries with skip lists and commutative hashing. Technical Report, Johns Hopkins Information Security Institute, 2000.
- [11]. W. Pugh. Skip lists: a probabilistic alternative to balanced trees. Commun. ACM, 33(6):668–676, 1990.
- [12]. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Contemporary Mathematics 324(1), 71–90 (2003).