Regular Paper

Communications of the CCISA
Vol. 26 No. 1 Feb. 2020

Verifiable Data Streaming via Noncryptographic Approach

Fan-Hsun Tseng [1*], Fan-Yi Kao [2]
[1, 2] Department of Technology Application and Human Resource Development, National Taiwan Normal University
[1] fhtseng@ntnu.edu.tw、[2] kao660021@gmail.com

Abstract

We propose a novel solution for verifiable data streaming via noncryptographic approach (NAVDS). Our proposed NAVDS is featured by its execution performance, compared to the existing solutions with significant computation burden on both client and server sides.

Keywords: Noncryptographic, Verifiable Data Streaming, Verifiable Data outsourcing

1. Introduction

The *verifiable data outsourcing* has always been an active research issues. In the conventional setting of verifiable data outsourcing, the client owns the entire set of data, and will be able to verify the query answer integrity after the data outosourcing. Nevertheless, in the big data era, the data of overwhelmingly high volume come with extremely high speed. This makes the client unable to keep all of the data in the local storage and then outsource them to the remote server. Therefore, a problem of verifiable computation in a steaming setting, where the client (verifier) outsources the numeric data in a streaming fashion to an untrusted server (prover), is considered in this paper. Such a *verifiable data streaming*, where the operations APPEND, QUERY, and VERIFY are available to use, is featured by the client being only able to ``see" the current element; the previous elements cannot be stored and the future elements are unpredictable. Despite the above constraint, the client is allowed to keep a small-sized local state. Under this setting, when the client issues a point query (*e.g.*, retrieving a particular data element) and receives the query result from the server, the cached state serves to verify the query result correctness. A conceptual illustration of verifiable data streaming is shown in Fig. 1, where the client can keep a local state for elements 2, 3, 4, 4, 2, 9 while the server may maintain a search structure (*e.g.*, B-tree) for answering queries. In this paper, we develop a novel and practical technique for verifiable data streaming with the particular focus on the verifiable point query.

* Author (Corresponding author.)

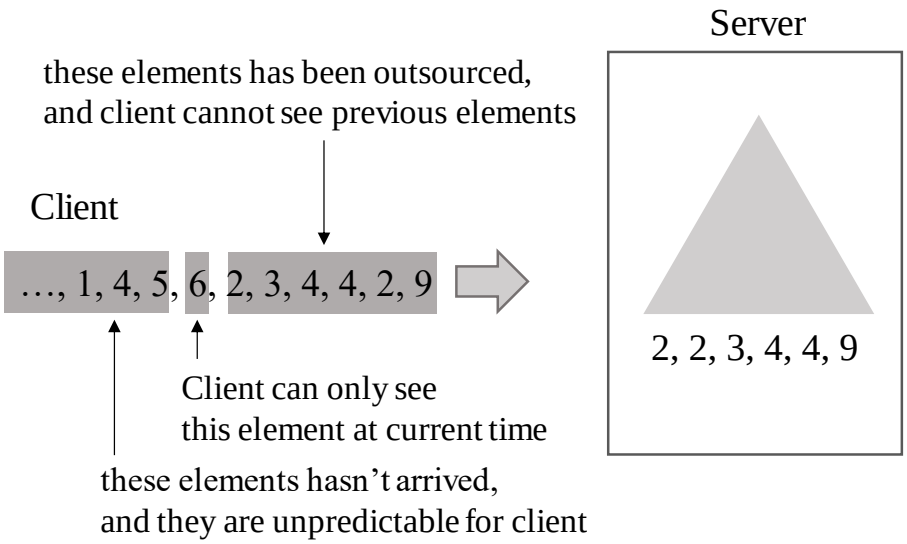

Figure 1: Conceptual illustration of verifiable data streaming and our solution.

Ordered neighborhood chain or Merkle tree [6] are common approaches to the conventional verifiable data outsourcing. In essence, the client keeps only the partial information about the chain or tree, which can be used for the verification purpose. Unfortunately, the existing techniques cannot apply to the streaming setting because they require the client to own the entire data. Hence, the primary challenge of verifiable data streaming lies in the design of the protocol with the update functionality.

2. Related Work

2.1 Verifiable Data Streaming

Verifiable data streaming has attracted the research attention recently. In particular, Schroder and Schroder [10] initiated the study of verifiable data streaming, where a stream s_1, s_2, … , of elements is processed and then outsourced to the server. Schroder and Schroder propose to use Chameleon Authentication Tree (CAT) to retrieve s_i (point query), $i \in [1, \alpha - 1]$, at the α-th time unit. In essence, the client in CAT still builds up a Merkle tree for the data stream but with the conventional hash function replaced by Chameleon hash function. The ordinary CAT method suffers from the limitation of bounded (or say, predefined) stream elements. The VeriStream [11] solution, where the CAT is used with a minor twist, is then proposed for the unbounded stream elements. Kim and Jeong [3] propose to employ ordinary binary hash tree to handle verifiable data streaming. Their idea is to store the data

elements in both internal and leaf nodes of the binary hash tree, in contrast to the data elements stored only in leaf nodes in CAT. Kim and Jeong claim to achieve $O(1)$ complexity for the storage, computation, and communications, but fail to prove it formally.

Yu [12] constructs an updatable Merkle tree by taking advantage of the conventional hash function replaced by simple arithmetic operations. In essence, in Yu's solution, once the new data element comes, the tree roots on both client and server sides will updated locally and immediately, except that the client can do the update in plaintext while the server can only do the update blindly (all of the data elements are encrypted homomorphically [2, 7]). Under the same setting, Papamanthou *et al*. [8] proposed Generalized Merkle Tree (GMT) to secure both the point and range queries. GMT can be thought of as an algebraic hash tree, with the hash function replaced by more complicated vector operations over the lattice to accomplish the local update of the client's state. Despite the performance improvement of GMT in [9], GMT still suffers from the performance inefficiency due to the intensive use of lattice-based hash function and projection function. Very recently, a nearly optimal solution [5] has been proposed; both the storage and communication reach $O(1)$ overhead. The solution in [5] uses cryptographic accumulator to store the revoked data elements. The client is then able to check whether the data element in the query response has been revoked previously.

2.2 Research Challenge

All of the existing solutions for verifiable data streaming suffer from performance issue. In particular, all of the solutions exploit the heavyweight cryptographic primitives such as Chameleon hash function, homomorphic encryption, and cryptographic accumulator. The common characteristic of these cryptographic primitives is the use of time-consuming modular exponentiation. As a consequence, we are looking for a lightweight solution for verifiable data streaming.

3. Proposed Method

Here, we propose a solution for verifiable data streaming via noncryptographic approach (NAVDS). The rationale behind NAVDS is the *dummy query*; more specifically, the client stores a small set of data elements in the local memory. The client occasionally issues the dummy point query to the server. Since the client will know the true answer of the dummy query from its local memory, when the server returns the falsified query response, the client may find the incorrect result.

There are a number of technical questions and concerns remained in the design and implementation of NAVDS; first, in the ideal case, the data elements in the client's local memory should be uniformly sampled from the data stream. Otherwise, for the positions more unlikely to be sampled, the server would be able to return the falsified answers of those positions without being detected. Second, the effectiveness of such dummy query-based solution relies on the fact that the server cannot differentiate dummy queries (*i.e.*, the queries that the client knows the answers) from genuine queries (*i.e.*, the queries that the client does not know the answer). Thus, it is necessary to make the time and queried positions of dummy and genuine queries follow the same distribution. Third, since the dummy queries will be generated, some of the genuine queries will be delayed. The delay for the genuine queries should be minimized.

3.1 Protocol Description

The APPEND operation for the client is shown in Fig. 2. Our proposed NAVDS maintains two buffers, M_1 and M_2 (shown in Fig. 3 and Fig. 4), which are used for caching the chosen data elements and caching the generated queries, respectively. In particular, our proposed NAVDS answers the first concern by taking advantage of reservoir sampling technique (Lines 1-6). In essence, the reservoir sampling is used to perform the uniform sampling over an infinite data stream. In addition, our design answers the second concern and ensures that all queries (both dummy and genuine) follow exponential distribution by keeping a buffer M_2 (Lines 7-9). We also assume a virtual proxy in the client, collecting dummy and genuine queries. This proxy will be formulated as M/G/1 queue, and perform the queueing analysis over the proxy to derive the query delay.

Algorithm: Append(s_i, M_1, M_2)
Input: s_i: *the coming data element*
M_1: *local memory for data elements*
M_2: *local memory for queriess*

```
1  do Reservoir-Sampling(s_i)
2  if s_i is sampled
3     if M_1 is not full
4        M_1 = M_1 ∪ s_i
5     else
6        Evict(M_1) and M_1 = M_1 ∪ s_i
7  if new query Q is generated
8     M_2 = M_2 ∪ Q
9  Issue-Query(M_2)
```

Figure 2: APPEND subroutine.

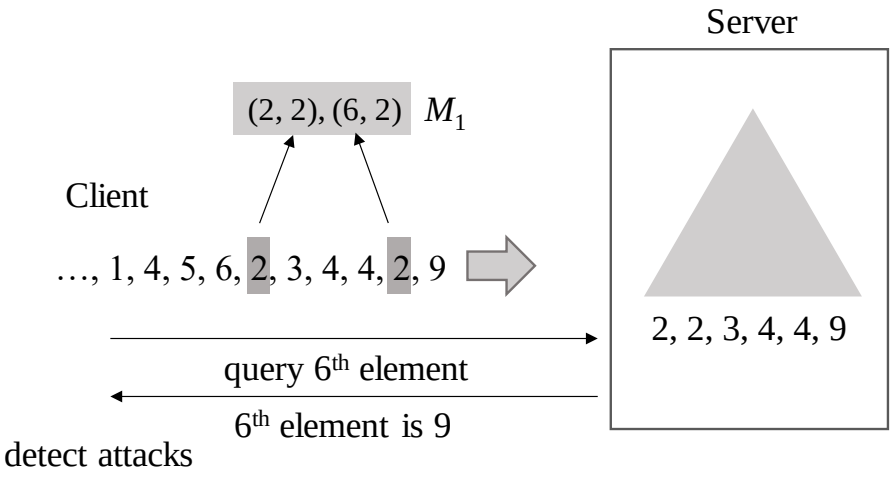

Figure 3: M_1 usage.

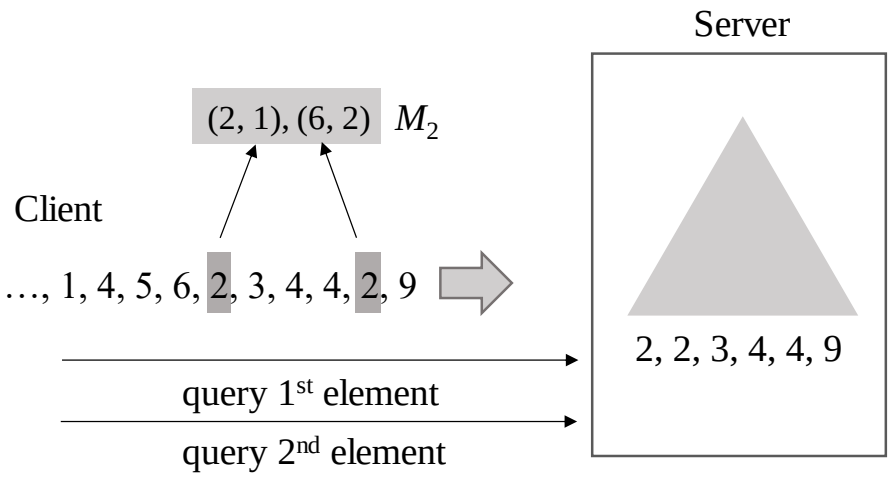

Figure 4: M_2 usage.

4. Conclusion

We propose a novel solution for verifiable data streaming via noncryptographic approach (NAVDS). Since our NAVDS does not involve any heavyweight cryptographic operations, it can achieve very high data rate and makes it very practical to be implemented.

References

[1] C. Gentry, ”Fully homomorphic encryption using ideal lattices,” *ACM STOC*, 2009.

[2] C. Papamanthou, E. Shi, R. Tamassia, and K. Yi, “Streaming authenticated data structures,“ *EUROCRYPT*, 2013.

[3] C.-M. Yu, “Lightweight Streaming Authenticated Data Structures,” *ACM CCS*, 2015.

[4] D. Schröder and H. Schröder, “Verifiable data streaming,” *ACM CCS*, 2012.

[5] D. Schröder and M. Simkin, “VeriStream - a framework for verifiable data streaming,” *FC*, 2015.

[6] H. Krawczyk and T. Rabin, ”Chameleon signatures,” *NDSS*, 2000.

[7] J. Krupp, D. Schröder, M. Simkin, D. Fiore, G. Ateniese, and S. Nuernberger, “Nearly optimal verifiable data streaming,” *PKC*, 2016.

[8] K. S. Kim and I. R. Jeong, “Efficient verifiable data streaming,” *Security and Communication Networks*, vol. 8, no. 18, pp. 4013-4018, December 2015.

[9] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” *EUROCRYPT*, 1999.

[10] R. C. Merkle, ”A digital signature based on a conventional encryption function,” *CRYPTO*, 1988.

[11] S. Benabbas, R. Gennaro, and Y. Vahlis, “Verifiable delegation of computation over large datasets,” *CRYPTO*, 2011.

[12] Y. Qian, Y. Zhang, X. Chen, C. Papamanthou, “Streaming authenticated data structures: abstraction and implementation,” *ACM CCSW*, 2014.