New Techniques for Streaming Authenticated Data Structures

Chia-Mu Yu[1,2], Shin-Ying Huang[3], Yennun Huang[3]
[1]Department of Computer Science and Engineering, Yuan Ze University
[2]Innovation Center for Big Data and Digital Convergence, Yuan Ze University
[3]Research Center for Information Technology Innovation, Academia Sinica

***Abstract*—We develop two novel techniques, FHMT and HWMT, for streaming authenticated data structures to achieve the streaming verifiable computation. By leveraging the computing capability of fully homomorphic encryption, FHMT shifts almost all of the computation tasks to the server, reaching nearly no overhead for the client. HWMT strikes the performance balance between the client and server via the proposed tree decomposition technique over the Merkle tree.**

I. Streaming Verifiable Computation

We consider the 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). A characteristic of such a *streaming verifiable computation* is that the client is only able to "see" the current element; the previous elements cannot be stored and the future elements are unpredictable. However, for a data stream, the client is allowed to keep a constant-sized local state. When the client issues a range query and receives the query result from the server, the cached local state can be used to verify the query result correctness. A conceptual illustration of streaming verifiable computation is shown in Fig. 1a, where the client can keep a local state for elements 2, 3, 4, 4, 2, 9 while the server may maintain a search structure for the outsourced elements (*e.g.*, B-tree). In this paper, we develop two novel techniques for *streaming authenticated data structures* to accomplish the streaming verifiable computation.

A common approach to the conventional authenticated data structure is for both the client and server to construct an ordered neighborhood chain or a Merkle tree [2]. 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 be in possession of the entire data stream. Hence, the primary challenge of streaming authenticated data structure lies in the design enabling the client to update the verification state locally.

II. Prior Art

There have been only few research efforts for security issues in the streaming setting. Schröder and Schröder [5] initiated the study of verifiable data streaming, where a stream s_1, s_2, ..., of elements is processed and then outsourced to the server. At the αth time unit, the client, via the Chameleon Authentication Tree (CAT), is able to retrieve s_i (point query), $i \in [1, \alpha - 1]$, with a proof from the server for the client's verification. In essence, the client in CAT still builds up a Merkle tree for the data stream but with the Chameleon hash function applied to each incoming element. Under the same setting, Papamanthou *et al*. [3] 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 [4], GMT still suffers from the performance inefficiency due to the intensive use of lattice-based hash function and projection function.

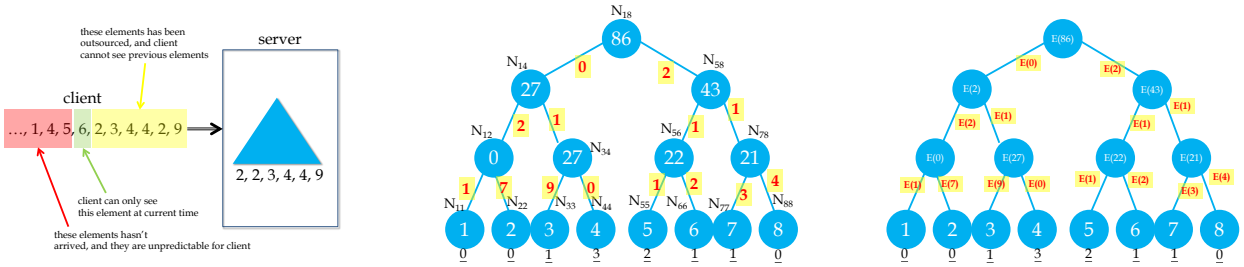
To reduce the operation cost, one may consider using only the basic arithmetic in the design of streaming authenticated data structures. As shown in Fig. 1b, our core idea is still to use Merkle tree with leaves from the universe $[1, M]$ of elements and with the hash function replaced by the linear combination of the labels of children nodes. Let N_{ij}^{α} be the label of the node with leaves exactly from N_{ii}^{α} to N_{jj}^{α} as the descendant nodes after the element s_α is considered. Assume that each edge between the parent node $N_{i_1 j_1}^{\alpha}$ and child node $N_{i_2 j_2}^{\alpha}$ is associated with a weight $w_{i_2 j_2}$ (yellow numbers in Fig. 1b). The client's local state consists of only the tree root N_{1M}^{α} and can be updated by simply calculating $N_{1M}^{\alpha} = N_{1M}^{\alpha-1} + w_{i_1 j_1} w_{i_2 j_2} \cdots w_{i_{\log M}, j_{\log M}} s_\alpha$, where $N_{i_{\log M}, j_{\log M}}^{\alpha} \rightarrow \ldots, \rightarrow N_{i_2, j_2}^{\alpha} \rightarrow N_{i_1, j_1}^{\alpha}$ constitutes the path from s_α to the root. The other procedures remain the same as in the conventional Merkle tree. Nevertheless, such a naïve method is flawed in that w_{ij}'s need to be publicly known, and therefore the server can find another set of coefficients to meet the same N_{1M}^{α}.

III. Proposed Methods

Motivated by the above flawed design, we propose two streaming authenticated data structures, Fully Homomorphic encryption-based Merkle Tree (FHMT) and Hash-Weighted Merkle Tree (HWMT). While the former incurs the asymmetric overhead by shifting the computation burden to the server, the latter trades the slightly increased communication burden for the better balanced efficiency for the client and server.

A. FHMT

Assume that the server knows $\mathcal{E}(w_{ij})$ for all edges, where $\mathcal{E}(\cdot)$ denotes the fully homomorphic encryption [1] and all of the w_{ij}'s can be generated from a master key possessed by the client only. When receiving s_α, the client updates $N_{1M}^{\alpha} = N_{1M}^{\alpha-1} + w_{i_1 j_1} w_{i_2 j_2} \cdots w_{i_{\log M}, j_{\log M}} s_\alpha$ and sends



(a) Streaming verifiable computation (b) Naïve solution (c) A FHMT example.

Fig. 1: Conceptual illustration of streaming verifiable computation and solutions.

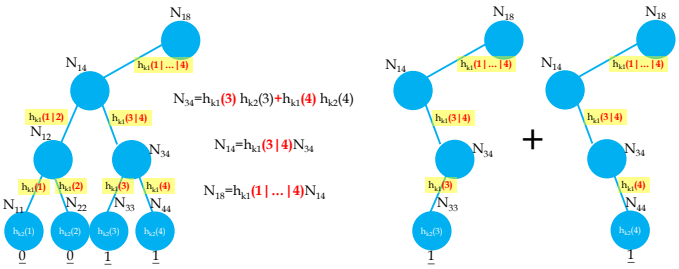
only s_α to the server. After receiving s_α, the server first calculates $\mathcal{E}(s_\alpha)$. Then, the server is able to update the relevant internal nodes based on the full homomorphism property of $\mathcal{E}(\cdot)$. In particular, the server calculates the encrypted labels of the root and relevant internal nodes, $\mathcal{E}(N^\alpha_{1M}) = \mathcal{E}(N^{\alpha-1}_{1M}) \oplus (\mathcal{E}(w_{i_1 j_1}) \otimes \mathcal{E}(w_{i_2 j_2}) \cdots \otimes \mathcal{E}(s_\alpha)) = \mathcal{E}(N^{\alpha-1}_{1M} + w_{i_1 j_1} w_{i_2 j_2} \cdots s_\alpha)$ and $\mathcal{E}(w_{i_\pi j_\pi}) = \left(\bigotimes_{i,j=1}^{\pi} \mathcal{E}(w_{ij})\right) \otimes \mathcal{E}(s_\alpha)$, $\pi = 1, 2, \ldots, \log M$, where $\oplus$ and $\otimes$ denote the homomorphic addition and multiplication. An example of how FHMT works is given in Fig. 1c.

Consider a range query $[d_\ell, d_r]$, $d_\ell, d_r \in [1, M]$. The server not only returns all of the elements within $[d_\ell, d_r]$ but also the sibling paths [2] of the elements d_ℓ and d_r. The client, after receiving the query result and sibling paths, calculates the tree root according to w_{ij}'s, sibling paths, and the returned elements. The query result is considered correct if there is a match against the cached tree root.

One can see that the computation burden is largely shifted to the server. During the update, the client only needs to perform $\log M - 1$ scalar multiplication and one scalar addition operations. All of the costly operations such as the tree update in the encryption domain are executed by the server. On the other hand, from the security point of view, the failure to offer the authentication guarantee in the naïve method comes from the fact that different sets of coefficients may imply the same root. Nevertheless, the weights in FHMT are unknown to the server, making the server unable to find another possible linear combination.

B. HWMT

Let k_1 and k_2 be secret keys owned by the client only. Assume that the server only knows $h_{k_1}(w_{ij})$ for all w_{ij}'s, where $h_k(\cdot)$ denotes a HMAC. The server initiates an empty Merkle tree. When the client receives s_α, it calculates $h_{k_2}(s_\alpha)$. Then, the client updates the tree root, $N^\alpha_{1M} = N^{\alpha-1}_{1M} + h_{k_1}(w_{i_1 j_1} h_{k_1}(w_{i_1 j_1} \cdots h_{k_1}(w_{i_{\log M} j_{\log M}}) h_{k_2}(s_\alpha)$, and all of the internal nodes along the path from s_α to the root, $N^\alpha_{i_\pi j_\pi} = \left(\prod_{i,j=1}^{\pi} h_{k_1}(w_{ij})\right) \cdot h_{k_2}(s_\alpha)$, $\pi = 1, \ldots, M$. After that, the client sends $\left(\prod_{i,j=1}^{\pi} h_{k_1}(w_{ij})\right) \cdot h_{k_2}(s_\alpha)$ and $h_{k_2}(s_\alpha)$ to the server. Afterwards, the server updates the nodes on the corresponding path accordingly. An example of HWMT is shown in Fig. 2a. Since HWMT can be decomposed into a simple combination of different root-to-leaf paths, the client with the knowledge of secret keys helps build up and send partial trees to the server. For example, as the outsourced elements are only 3 and 4, the HWMT in Fig. 2b in fact can be decomposed into two paths in Fig. 2b. Consider a



(a) A HWMT example. (b) Tree decomposition.

Fig. 2: Conceptual illustration of HWMT.

range query $[d_\ell, d_r]$. Similarly, the server not only returns all of the elements within $[d_\ell, d_r]$ but also the sibling paths of the elements d_ℓ and d_r. The client in possession of k_1 and k_2 recovers the tree root according to w_{ij}'s, sibling paths, and the returned elements. The query result is considered correct if there is a match against the cached tree root.

One can observe a slightly increased communication burden at the client side. Nonetheless, as the transmission of $\log M$ numbers would not be a considerable task in most of the cases, the sacrifice of the client's efficiency greatly reduces the server's overhead; now the server only needs $\log M$ simple additions to accomplish the update of HWMT. On the other hand, from the security point of view, similarly since the weights in HWMT are all unknown to the server, this makes the server unable to find another possible linear combination.

IV. Conclusion

Two novel streaming authenticated data structures, FHMT and HWMT, are proposed to achieve the streaming verifiable computation. Due to their efficiency, they are considered applicable for the systems with the resource-constrained client devices such as sensors, IoT devices, and mobile phones.

References

[1] C. Gentry. Fully homomorphic encryption using ideal lattices. *ACM Symposium on Theory of Computing (STOC)*, 2009.
[2] R. C. Merkle. A digital signature based on a conventional encryption function. *International Cryptology Conference (CRYPTO)*, 1988.
[3] C. Papamanthou, E. Shi, R. Tamassia, and K. Yi, Streaming authenticated data structures. *EUROCRYPT*, 2013.
[4] Y. Qian *et al.* Streaming authenticated data structures: abstraction and implementation. *ACM CCSW*, 2014.
[5] D. Schröder *et al.* Verifiable data streaming. *ACM CCS*, 2012.