Abstract

In the setting of streaming verifiable computation, a verifier and a prover observe a stream of n elements $x_1, x_2, \ldots, x_n$ and later, the verifier can delegate a computation (e.g., a range search query) to the untrusted prover over the stream. The prover returns the result of the computation and a cryptographic proof for its correctness. To verify the prover's result efficiently, the verifier keeps small local (logarithmic) state, which he updates while observing the stream. The challenge is to enable the verifier to update his local state with no interaction with the prover, while ensuring the prover can compute proofs efficiently.

Papamanthou et al. (EUROCRYPT 2013) introduced *streaming authenticated data structures* (SADS) to address the above problem. Yet their scheme is complex to describe and impractical to implement, mainly due to the use of Ajtai's lattice-based hash function. In this work we present an *abstract* SADS construction that can use any hash function satisfying properties that we formally define. This leads to a *simpler* exposition of the fundamental ideas of Papamanthou et al.'s work and to a *practical* implementation of a streaming authenticated data structure that employs the efficient SWIFFT hash function, which we show to comply with our abstraction. We implement both the EUROCRYPT 2013 construction and our new scheme and report major savings in prover time and public key size.