Optimizations and Discoveries Regarding Constant-Communication Oblivious RAM

In today's world, ever-increasing amounts of data are being stored online in external, untrusted servers - the "cloud". Considering the importance of such data to users, businesses, and organizations, it is necessary to protect data from mishandling and exploitation on those servers. While current encryption schemes can protect the actual data on a server from being read, a user's access pattern to files on a server is vulnerable. These metadata can be analyzed by malicious servers, and potentially used to gain information on the actual data. Oblivious RAM (ORAM) is a cryptographic primitive that hides a user's access pattern metadata from untrusted servers. Our goal in this work is to optimize the latest tree-based ORAM algorithm, Constant Communication ORAM (C-ORAM), using a simulation of the eviction process. We improve C-ORAM by conducting runtime tests as well as by determining the optimal values of parameters such as eviction frequency and bucket size - two important factors in the C-ORAM algorithm. To determine these values using experimental analyses, we implement the C-ORAM eviction cycle using Python 2.7. Using results from our experimental analyses, we are able to optimize the C-ORAM algorithm. We also provide an intuitive security explanation as well as an optimization for an essential operation in C-ORAM. These advances are useful for the further development of Oblivious RAM.