Optimizations and Discoveries Regarding Constant-Communication Oblivious RAM

In today's world, an ever-increasing amount of data is being stored online in the "cloud" - external, untrusted servers. Considering the importance of such data to users, businesses, and organizations, it is necessary to protect data from mishandling and exploitation from 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. This metadata can be analyzed by malicious servers, and potentially used to gain information about the actual data. Oblivious RAM (ORAM) is a cryptographic primitive that hides a user's access pattern metadata from untrusted servers. In this work, we discuss and analyze various ORAMs. We first provide an explanation of the various tree-based ORAM algorithms, including Constant Communication ORAM (C-ORAM), a constant-bandwidth, high-efficiency algorithm. In our project, we seek to improve upon and complement C-ORAM by conducting runtime tests as well as 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 tests, we implement our own version of the C-ORAM eviction cycle using Python 2.7, while highlighting certain conditions necessary to successfully run C-ORAM. Using our tests, we are able to determine the optimal values of eviction frequency and bucket size while ensuring that the algorithm does not “overflow” - or crash. These advances are useful to the development of the ORAM algorithm, especially C-ORAM.