Meenakshi Nagarajan

Constants Count: Practical Improvements to Oblivious RAM

Research problem and its importance

Encrypting the data in a cloud platform does not ensure complete privacy. Analyzing access pattern on the cloud to understand the data transferred between client and server may affect privacy. This will provide the attacker information on private files and data in remote location. ORAM helps users in maintaining privacy of user's data against memory access privacy leakage. There are many implementations of ORAM. ORAM technique poses a significant limitation on client storage and bandwidth. Reducing program specific client's storage and number of additional messages communicated with the server to obtain the data are the key improvements provided in this paper.

Technical challenges of the problem

In this paper, Ring ORAM implementation's output is proposed. In Ring ORAM, data is stored in nodes and sequence in a tree model. Reducing the number of additional messages communicated to the server, bucket size to store temporary data and building a scalable solution for increasing the bucket size are the key technical challenges addressed in this paper.

Proposed approach

Ring ORAM is a tree based method where server storage is a binary tree of buckets and data is spread across multiple levels. Client storage implementing this scheme must keep track of root to leaf nodes. All the higher-level nodes will route to the leaf node to access the data. Eviction of data, path to be traversed and read are maintained at the client end. Access mechanism is broken into steps like Position Map lookup, Read Path, Evict Path, Early Reshuffles. With this approach, online bandwidth is reduced by 60 times for simple computations and performance is improved by 1.5 times as compared to PATH ORAM.

Strengths of the approach

1. More Efficient Bandwidth usage.
2. Works on Small Client Storage setting
3. Doesn't depend on any specific bucket sizes like traditional approaches.
4. Can easily scale for larger client storages.

Possible weaknesses

1. ORAM has negative impact on the performance database due to operational overheads.
2. Complex operations like joins etc. leads to additional operational overheads.
3. Achieving concurrency on large server data is inefficient using ORAM.

References:

1. Oblivious RAM: A Dissection and Experimental Evaluation