Advanced Topics in Online Privacy and Cybersecurity (67515), Spring 2025

Assignment 1: ORAM

Introduction

Your task in this assignment is to implement a storage application that hides client access patterns from a remote server - otherwise known as an ORAM. The specific variant of ORAM that you are required to implement is Path ORAM first mentioned in [1]. You are required to implement the non-recursive Path ORAM variant.

Programming task (functionality)

You are required to implement a client and server.

Definitions:

- **id** - an integer
- **data** - a string with 4 characters

Requirements:

- Server is initialized to support N datablocks
- The client can store data associated with an ID on the server by calling **self.store_data(server, id, data)**
- Given a name, a client can retrieve associated data by calling by calling **self.retrieve_data(server, id),** the client should return None if the data does not exist.
- Client can delete data associated with an ID from server by calling **self.delete_data(server, id, data)**
- The client has access to O(NUMBLOCKS) storage used mainly for the stash (data is stored on the server). You are welcome(but in no way required) to implement the recursive PATH ORAM and use a smaller stash.

Note that the server is the owner of the data storage, and should implement the methods described in [1] to interact directly with said data storage in an oblivious manner.

The above API was written so that you wouldn't need to deal with network communication between the client and server and to emphasize what functionality is expected of you to implement. The server should store the tree structure, and there is no defined API for it. When in doubt about the algorithms you are expected to implement in the server, consult the paper.

Programming task (security)

The server should be oblivious to data content and the client's access patterns. This should be accomplished by use of end to end encryption and PATH ORAM.

The server should be unable to trick the client into accepting corrupt or outdated data (data integrity). This should be accomplished by use of authentication.

You may use this library for encryption and verification.

Submission Guidelines

Submit your code in **Python** along with a pdf design document. In the design document describe the architecture you built and how it satisfies the above requirements. Additionally, include the following performance benchmarks:

- Throughput (number of requests/sec) vs. N (DB size)
- Latency (time to complete a request) vs. throughput

Make sure to write a brief discussion of your benchmarks, mention how many runs of your ORAM you averaged to receive your results, and state the units you are using clearly.

Does your implementation benefit from multicore?

Good Luck!

References

[1] Emil Stefanov, Marten Van Dijk, Elaine Shi, T.-H. Hubert Chan, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: An extremely simple oblivious ram protocol. *J. ACM*, 65(4), apr 2018.