PathORAM - Design Document

Lior Ashkenazi | 315533059

April 27, 2022

1 Preface

This design document intends to introduce the PathORAM project - a project compromised of two compontents: the $client$ and the $server$, which communicates with each other by TCP/IP protocol to transfer files.

2 Client

The $client$ module compromised by 4 sub-modules:

- cloud
- crypto
- storage
- ui

As well more 3 helper files - config.py, data.py, log.py: config.py has some global constants which the $client$ module use, data.py has functions whose purpose is to handle files and log.py configures the logging of the client program, i.e., the informative messages the client will be shown during activation of the program.

The $client$ module process of sending a file is as follows:

$$ui$$

$$\downarrow$$

$$storage$$

$$\downarrow$$

$$crypto$$

$$\downarrow$$

$$storage$$

$$\downarrow$$

$$\begin{tikzcd} cloud \arrow[d] \\ \boxed{server} \end{tikzcd}$$

and the process of receiving a file is as follows:

$$\begin{tikzcd} ui \arrow[d] \\ storage \arrow[d] \\ cloud \arrow[d] \\ \boxed{server} \arrow[d] \\ cloud \arrow[d] \\ storage \arrow[d] \\ crypto \arrow[d] \\ storage \end{tikzcd}$$

2.1 cloud

- __init__.py
- cloud.py
- exceptions.py
- server_connection.py
- utils.py

Responsible for the communication with server, getting the data from other sub-modules and sending it to the server.

2.2 crypto

- __init__.py
- aes_crypto.py
- exceptions.py
- key_map.py
- utils.py

Apply all the cryptography of the program. For encryption (and decryption), the sub-modules uses:

- AES (256-bit key length)
- CBC-mode with IV(128-bit)

And lastly padding with PKCS7.
Moreover, the sub-module uses HMAC (256-bit key length) with SHA256 for authentication. Furthermore, the sub-module uses a key-map for storing for each password (currently, the program can only have one password!) a map of the pertained keys (specifically - Salt, AES and MAC) for future encryptions and decryptions.

2.3 storage

- __init__.py
- controller.py
- data_file_map.py
- exceptions.py
- file_processor.py
- oram.py
- stash.py
- tree_map.py
- utils.py

The core of the *client* module and perhaps of the entire program. Composed by controller.py which oversees the entire module operation, stash.py which resembles stash and oram.py which is a PathORAM tree. Furthermore, the module has maps for the PathORAM tree, for tagging each data block to a leaf in the tree, and for data blocks - note that we tag each data block with a data ID, the sub-module doesn't work with files per say, it works with data files, which are chunks of the files ther user uploads. For createing such chunks the file_processor is used for. Also, for combining chunks and then for dismantling them, the data_file_map is necessary - for padding and upadding if necessary (almost always it **is** necessary) and for saving information about expected file size.

2.4 ui

- __init__.py
- handler.py
- utils.py

Connected to the main.py (will be described later), is the bridge to the user. Handles every call of the user for a chosen action.

2.5 miscellaneous

- config.py
- data.py
- log.py

Described above.

3 Server

A sub-module which represents a simple server which uses TCP/IP protocol. Recieves and sends files and stores them in a data folder in $blockX.oram$ format. Unaware of the data in those files.

4 Benchmarking

The program send files one-by-one, so it support sending multiple files per one request. Moreover, multi-threading might be useful generally, but this program it doesn't as the program has one task per one request. Generally, a PathORAM program might benefit extensively using multi-threading and supporting sending and receiving multiple files simultaneously. Of course, a server which supports such thing is needed.

5 Instructions

For using the program, firstly one has to open a terminal in the PathORAM folder and activate the server in the following way:

```
python3 server/server.py
```

Now, for activating the client, one has to open a **new** terminal, while the server is online (!), and enter the following line:

```
python3 main.py
```

Thereafter, a new user has to enter a password (in the next activations, the client has a saved key_map file so the password is saved and has to be used in the next time). After entering a password, intructions for using the program are shown.

IMPORTANT NOTE For resetting the program, one has to delete the following folders from the $client$ module:

- stash
- cloud.map
- file.map
- key.map
- position.map

and may also delete oram.log.

Good Luck!