**SEC Project Report**

**Group 2 Alameda**

* **Daniel Trindade 76349**
* **Fernando Marques 76419**
* **Ricardo Abreu 76370**

System design:

The FS is composed of one file per user. Each file has one Public Key Block (PKBlock) and can have several Content Blocks (CBlocks). The PKBlocks are recognized by an id which corresponds to the hash of the public key of the user. These blocks are also signed for security reasons (explained further ahead). They also contain a byte array which only contains metadata. This metadata corresponds that file’s CBlocks’ ids (which are made up of 32 bytes each). These ids are stored in the order in which the corresponding CBlocks were added to the file.

The CBlocks are only composed by a byte array of content with a fixed size of 2048 bytes.

These blocks are recognized by an id which corresponds to the hash of its content. The CBlocks are sent to the server with extra information in their content which corresponds to the file id that each block belongs to. The server then filters the "real" content of the block from the id and with that it is possible to separate the content blocks per file.

The write function starts by requesting the PKBlock for a file so that it is possible to fetch the metadata for that specific file.

From the metadata it is possible to determine the amount of blocks there are, the amount of information (the number of CBlocks \* the fixed size) and more. This metadata also helps deciding if the write is occurring in an already written position forcing an overwrite or if it's beyond the end of file in which new blocks need to be created. The CBlocks are padded with zeros when the information is written in a position not yet available on the file (when the position exceeds the file size) and are also padded with zeros when the information isn't enough to fill the rest of the block.

As for the read it will also fetch the PKBlock first and from the metadata it is able to detect if the position to read is beyond the EOF or if it exceeds the EOF while reading. In this last case it will stop reading and return what it read until that moment of time.

Security Design (integrity and non-repudiation guarantees):

We used the SHA-256 algorithm to generate the hashes - this includes the hash of the CBlocks content and the hash used in the signature for the PKBlock. To generate a signature we used the RSA algorithm. The hash of the Cblocks’ content allowed us to verify if the content had been tempered in any way since H(X)!=H(Y) for every X!=Y. As for the PKBlock we signed it so we can assure that whoever is writing to that block is indeed its owner. Additionally, as the PKBlock stores the ids of the CBlocks, we can also guarantee that these ids are in fact the ones created from the CBlocks that the legitimate owner wrote – recall that the ids of a CBlock are the hash of its content.