Highly Dependable Systems

Project 1: File server with integrity guarantees

Group 15

70613 - Nuno Nogueira, **70638** - João Sampaio, **73991** - Pedro Braz

FS design

We implemented a file system providing various different dependability guarantees. In terms of stored data, there are two data types the DataBlock and the PKBlock.
In our approach the user's file is split among multiple DataBlock, and a single PKBlock that contains a list of the identifiers of all the DataBlocks belonging to the user file. By splitting the file's data among several blocks we eliminate the need to compute the entire file hash in cases where only a small portion of the file was added or edited, for example, in the event of a file with multiple Gigabytes, this approach allows the system to only compute the hash of the required blocks and not of the entire file. In Figure 1 an overview of the system data structures is shown.

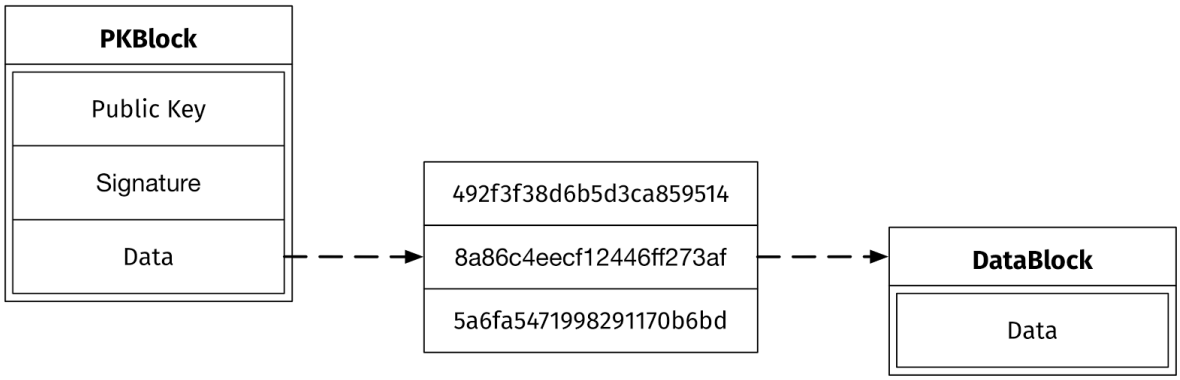

Figure 1 - System data structures overview.

One of the system requirements is to sign the user file using digital signature, we decided to only sign the PKBlock since it has the list of all the DataBlocks' hashes of the file, eliminating the need to digitally sign the entire file.
Between the client and server we use HTTP messages with JSON format to transfer the application data. We chose to implement an HTTP REST API in the block server as it is an easy way to achieve remote communication between the client library and the block server. The JSON format was picked as there are third-party libraries that allow the conversion of Java objects into JSON and vice-versa. By using one of these libraries we can work on a Java object in the block server and easily transfer it and keep working on it in the client library.

Integrity guarantees provided
Since we aim to provide integrity guarantee when we read a user's file, we compare the DataBlock identifier (hash) to the returned hash of the returned content, if the two match then we are certain it is the data we requested and it has not been changed.

Our system provides non repudiation guarantee, since the user digitally signed the PKBlock and the PKBlock contains the list of all the DataBlock hashes of the user file, then if a DataBlock hash is present in the user PKBlock then we are sure that the user placed that specific DataBlock in the file. As long as the signature verification checks out then the author cannot repudiate the contents of the file since it was signed using the author's unique private key.

Dependability guarantees provided
Besides the integrity guarantee previously stated, our system provides safety guarantee, the system will not disrupt the operation of other systems nor compromise the safety of any user operations. In the event of failure where the client library cannot continue to operate, certain exceptions are thrown in a safe manner.

Our Webserver is able to support multiple concurrent users, writing to different files.

Threat Model
Our system is able to guarantee file integrity under the following scenarios:

- **Man-in-the-middle attacks** where the data is intercepted between the client-server communication;
- **Server-side data tampering**;
- **Unauthorized clients writing to the data.**

The following tests are performed on the system to evaluate the correct detection of integrity failures:

- **Receive invalid hash on put_h** - The client calls the server with put_h to insert a new block remotely. The server will respond with an invalid identifier, as if the block's data was changed. The client needs to invalidate the write;
- **Receive invalid data on get -** The client calls the server with get to access a remote data block. This block's data been changed and it won't correspond with the hash given by the client;
- **Receive invalid hash on put_k** - The client attempts to access a remote public key block. This block's public key has been changed, and won't correspond with the client's public key hash;

- **Receive 400 while reading a public key block -** The server receives a put_k request with an invalid signature. The server must return a 400 status code;

- **Find a wrong signature while reading a public key block -** A client reads a public key block from the server. The data has been changed and the signature will be invalid.