Highly Dependable Systems

Project 3: Block Server with intrusion-tolerant replication

Group 15

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

New Features

- **Byzantine Quorum Consensus -** Read and write operation follow the design of a (1,N) Byzantine Regular Register, for 1 writing client and multiple readers. Client content is replicated across N server replicas, tolerating f byzantine failures, where N > 3f. Our approach is designed for 1 failure, with N being 5.
- **Message Authentication -** As a prerequisite for the quorum algorithm, communication must be correctly authenticated. Using the hmac algorithm, we allow hosts to verify the messages' origin.

Integrity guarantees provided

The quorum algorithm used is an **Authenticated-Date Byzantine Quorum** and provides **termination** and **validity** to any operation for at least 1 byzantine replica. For this limitation, we assure that any operation will complete, that a read will return the newest value and a read concurrent with a write will return the last value or the value being written.

A client's public block write is replicated across all *N* replicas, and the client needs to wait for 4 replicas ($(n+f) \div 2$) to return their acknowledgments. If *f* is 1, then we can confirm that at least 3 correct replicas, updated their state.

For a public block read, the client will wait for 4 replicas to return the correct block. From any 4 replicas, the client can always intersect a correct process, which has the newest block. Verification is easy because these blocks are signed, the block with the highest timestamp is the correct one, and the byzantine server can't interfere, incorrect

Content blocks are immutable, a replica has a block or doesn't and if the client created it previously, a correct process has it. A write only needs to wait for $f+1$ servers, this is because during a read, the server only waits for 1 process, discarding all responses with incorrect block hashes.

For a correct byzantine quorum, the use of Authenticated Perfect links is a requirement. This algorithm assures reliable delivery, no duplication and authenticity.

Because our server communication uses HTTP, we use TCP as the transport layer protocol, guaranteeing that correct messages are eventually delivered and any duplicated message isn't passed to the application. The only requirement left is authenticity. We add an HMAC to each message so that receivers can confirm the message's original sender.
In order to compute the HMAC, servers and clients have a predefined shared secret. The input data for the HMAC hash function is the concatenation of following fields in the header:

- **Content-Type;**
- **Request Method;**
- **Request Path;**
- **Session ID -** custom value to safeguard against replay attacks;

Threat Model

Our updated system is able to guarantee file integrity under new scenarios:

- **Replay attacks** where an outdated public key block is resubmitted to the server by a malicious user in place of the original client, overwriting the updated data. And where a client reads from a public key block, which is replaced by outdated data.

- **Invalid Certificate** Where a client submits an invalid certificate for its key, and the client's identity can't be verified.

The following tests are performed on the system to evaluate the implementation of the regular register algorithm:

- **Old PKBlock** - when the client request a PKBlock and one of the servers responds with older data, the client is able to detect this and uses the PKBlock returned by other servers.

- **Timeout Writes** - when more than 1 server doesn't respond to the request, due to failure, the consensus about the data state cannot be achieved and therefore the client launches an exception;

- **Authentication test** - in order to test replay attack protection, we created a test case where bad servers try to fake a response.