

# Project1 Report

## System Architecture

### Components

#### 1. Client (BlockStorageClient)

- Encrypts files using a symmetric cipher (e.g. AES/GCM or AES/CBC + HMAC).
- Divides each file into 4096 bytes blocks.
- Sends each encrypted block to the server with associated keyword tokens.
- Maintains a local index mapping files → block IDs.
- Supports keyword-based search and automatic file retrieval.
- Verifies data integrity and deletes files from the server upon successful download.

#### 2. Server (BlockStorageServer)

- Receives, stores, and indexes encrypted blocks.
- Persists metadata associating block IDs with keyword tokens.
- Supports commands from clients (store, list, search, get, delete).
- Deletes data securely upon client request.

#### 3. Crypto Layer (CryptoStuff + CryptoConfig)

- Handles encryption, decryption, key management, and HMAC generation.
- Loads or creates cryptographic keys from a secure keystore ( `clientkeystore.jceks` ).
- Supports configurable algorithms via `cryptoconfig.txt` .

---

## Cryptographic Design

### Configurable Crypto Parameters

All cryptographic parameters are defined in `cryptoconfig.txt` , for example:

```
CIPHER = AES/GCM/NoPadding
KEYSIZE = 256
HMAC = HmacSHA256
MACKEYSIZE = 256
```

These settings are loaded at runtime by `CryptoConfig.java` .

### Encryption Modes

Even if both CIPHER and HMAC are defined in `cryptoconfig.txt` , if the cipher is AEAD, it won't use the HMAC.

The system supports:

- **AES/GCM and CHACHA20-Poly1305 (AEAD)**: Provides both confidentiality and integrity.
- **AES/CBC + HMAC**: When AEAD is not available, integrity is ensured via HMAC.

## Key Management

- Keys are securely stored in a Java KeyStore (JCEKS) file.
- On first use:
  - A **new key** is generated and stored as `"clientKey"`.
  - A **new HMAC key** is generated and stored as `"clientHMACKey"`, if the user chooses to use HMAC.
  - A **new HMAC key** is generated and stored as `"keywordKey"`, used to generate the tokens for the keywords.
- On subsequent runs, keys are retrieved from the keystore using the user password.

## Keyword Tokenization

Each file can be associated with searchable keywords.

Keywords are never sent in plaintext. Instead, they are transformed into secure HMAC-based tokens:

```
token = HMAC(clientHMACKey, keyword)
```

These tokens are stored on the server and used for keyword-based search without revealing the actual keywords.

## File Upload (PUT Command)

When the client executes `PUT`:

- The file is read and split into 4096 bytes blocks.
- Each block is **encrypted** using the key.
- A **SHA-256 hash** of the encrypted block is computed to generate a unique block ID.
- The first block includes **keyword tokens**, computed via `HMAC(clientHMACKey, keyword)`.
- Each block is sent to the server with:

```
STORE_BLOCK
<blockId>
<blockLength>
<encryptedData>
[<keywordTokens>]
```

- The server writes each block to the `blockstorage/` directory and updates its metadata.
- Once the upload completes, the original file is deleted from the client.

## File Retrieval (GET Command)

### Retrieval by Filename

```
cltest GET <filename> <output_directory>
```

- The client reads the local index to find all block IDs for the file.
- It sends a `GET_BLOCK` command for each block.

- Each block is decrypted locally using the AES key and verified.
- If all blocks are valid, the client sends `DELETE_BLOCKS` to remove them from the server.
- The decrypted file is written to the specified output directory.

## Retrieval by Keyword

```
cltest GET <keyword> <output_directory>
```

- The client generates the token for the keyword.
- Sends a `SEARCH` command to the server.
- The server returns all block IDs containing that keyword token.
- The client maps each block ID to a local filename via `client_index.ser`.
- All matching files are downloaded and reconstructed in the chosen directory.

## File Search (SEARCH Command)

```
cltest SEARCH <keyword>
```

- The client generates a HMAC based token for the given keyword.
- It sends this token to the server using the `"SEARCH"` command.
- The server searches its metadata for blocks associated with that token and returns their block IDs.
- The client maps these block IDs back to filenames using its local `client_index.ser` and prints the results.

## File List (LIST Command)

```
cltest LIST
```

- The client sends a `"LIST_BLOCKS"` command to the server.
- The server responds with all stored block IDs.
- The client checks each block ID against its local index to identify which files they belong to.
- Each unique filename is printed.

## Integrity Verification

Integrity is always checked during decryption:

- In **AES/GCM mode**, the GCM authentication tag is verified.
- In **AES/CBC + HMAC mode**, a separate HMAC is computed and validated.

If any block fails integrity verification, the client:

- Prints a warning message.
- Does not delete the corresponding blocks from the server.