**RESULTS AND DISCUSSION**

**A. Performance Analysis of Upload and Download**

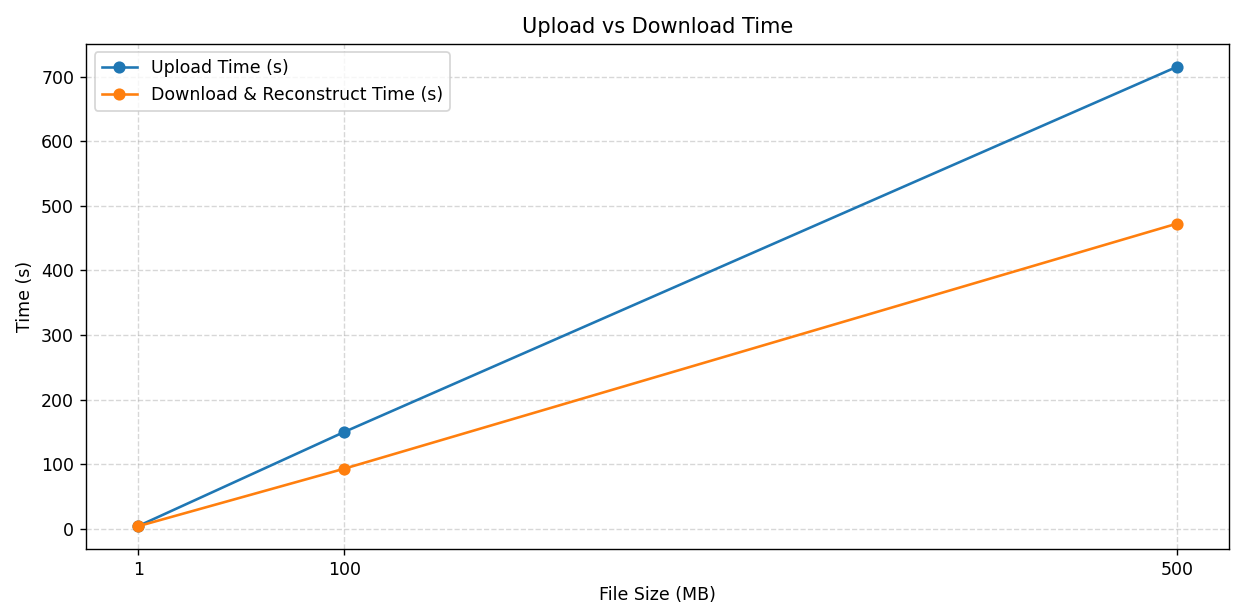
To find the efficiency of uploading a file to the Dropbox using our method. We have tested with different file sizes (1 MB, 100 MB, and 500 MB) to calculate the time and Speed taken to upload and download the file. The calculated results are shown in Table I

**Table I – Performance Summary of Encrypted Cloud Upload and Retrieval**

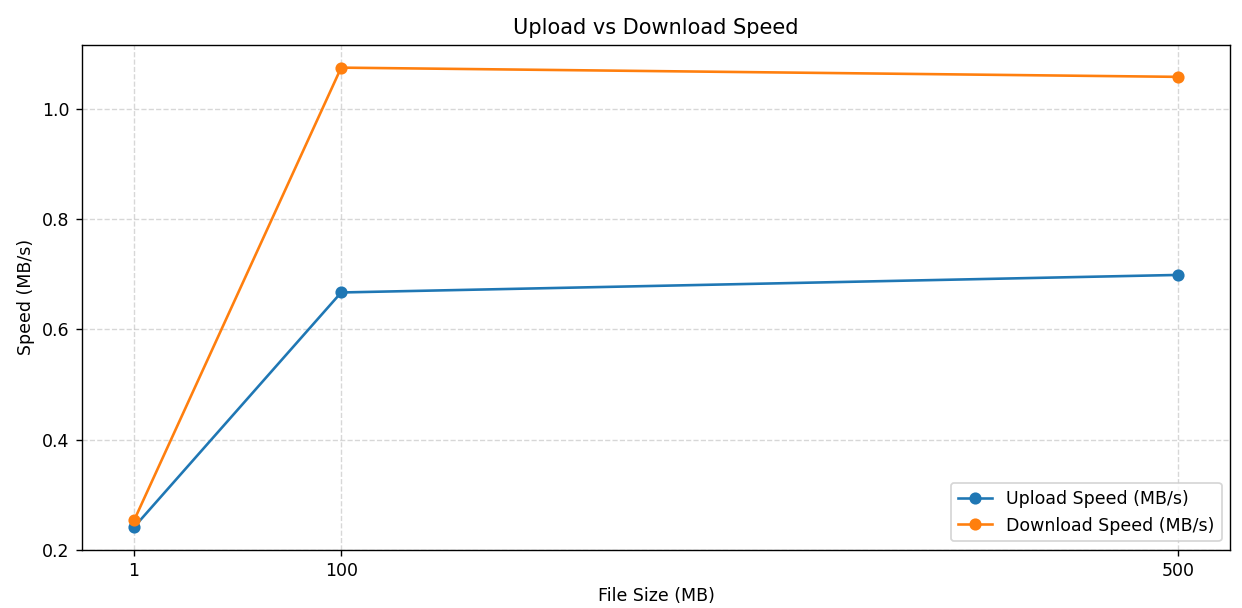
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **File Size** | **Fragments** | **Upload Time (s)** | **Upload Speed (MB/s)** | **Download & Reconstruct Time (s)** | **Download Speed (MB/s)** |
| 1 MB | 1 × 1 MB | 4.1446 | 0.2413 | 3.9443 | 0.2535 |
| 100 MB | 10 × 10 MB | 149.9806 | 0.6668 | 93.0743 | 1.0744 |
| 500 MB | 50 × 10 MB | 715.7653 | 0.6986 | 472.7407 | 1.0577 |

**Observations:**

It is observed that the time taken to upload a file increases as the file size increases, i.e., ~0.7 MB/s, and the download time is comparatively less than the upload time, i.e., ~1 MB/s. This shows that retrieval operations are comparatively faster than uploads.



**Fig. 1 – File Size vs Upload & Download Time**



**Fig. 2 – File Size vs Download & Reconstruction Speed**

**B. Encryption Overhead of AES–ECC**

We examined both AES-GCM encryption and ECC-based key encapsulation separately and evaluated the cryptographic overhead relative to the total upload latency, and the results are shown in table II.

**Table II – AES & ECC Overhead Contribution**

|  |  |  |  |
| --- | --- | --- | --- |
| **File Size** | **AES Encrypt Time (Total)** | **ECC Key Encrypt Time** | **% of Total Upload Time** |
| 1 MB | 0.0039 s | 0.0129 s | <0.1% |
| 100 MB | 0.3757 s | 0.0137 s | <0.3% |
| 500 MB | 2.2991 s | 0.0135 s | <0.3% |

**Conclusion:**  
From this, It can be concluded that the overhead for ensuring security remains negligible. This is further illustrated by the fact that latency attributed to cryptography stands at an inconsequential fraction of 0.3% and this goes to show that the network transfer is the bottleneck and not the encryption.

**C. Fragment Loss and Data Confidentiality**

For the purpose of checking robustness, a few random cloud-archived encrypted fragments were retrieved prior to deconstruction. Decrypting these was expected to end with

**DownloadError: path/not\_found.**

This confirms that if only the AES key is accessible, recovery will not succeed, meaning it is impossible to retrieve anything if not all fragments are available. This explains the system default implementation of the all-or-nothing security robustness paradigm.

**D. Storage and Network Overhead Analysis**

As fragmentation occurs, there is a little more overhead in terms of metadata and authentication. Each AES-encrypted fragment has a little more than 0.02% expansion because of an Addition Initialization Vector (IV) and GCM(Galois/Counter Mode) authentication tag. The manifest file, which contains the fragment index and the ECC-wrapped AES key, increases in size by 7–14 KB for each upload.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Advantage** | **Cost** |
| Fragmentation | Increases parallelism and security isolation | More API calls (latency) |
| AES–ECC Hybrid | Guarantees strong confidentiality & key isolation | Negligible additional cryptographic header overhead |