**Time Taken to upload and download:**  
  
**1mb**=== Performance Summary (Upload) ===

File: test\_1MB.bin

Original size: 1048576 bytes (1.00 MB)

Fragments: 1 (fragment\_size=1048576 bytes)

Read time: 0.0342 s

Split time: 0.0000 s

ECC (AES-key encrypt) time: 0.0129 s

Total AES fragment encryption time (sum): 0.0039 s

Avg encrypt per fragment: 0.003936 s

Total fragment upload time (sum): 2.8871 s

Avg upload per fragment: 2.887113 s

Total uploaded bytes (fragments only): 1048608 bytes (1.00 MB)

Manifest size: 8162 bytes

End-to-end total time: 4.1446 s

Effective upload throughput (original file): 0.2413 MB/s

Effective upload throughput (uploaded bytes incl. IVs): 0.2413 MB/s

=== Performance Summary (Download & Reconstruct) ===

File: test\_1MB.bin

Original size (manifest): 1048576 bytes (1.00 MB)

Fragments: 1

AES-key ECC decrypt time: 0.0134 s

Total download time (sum): 2.2821 s

Avg download per fragment: 2.282073 s

Total AES fragment decrypt time (sum): 0.0032 s

Avg decrypt per fragment: 0.003220 s

Total downloaded bytes (fragments): 1048608 bytes (1.00 MB)

End-to-end total time (download+decrypt+merge): 3.9443 s

Effective reconstruction throughput (original file): 0.2535 MB/s

Effective reconstruction throughput (downloaded bytes): 0.2535 MB/s

**100mb**

=== Performance Summary (Upload) ===

File: test\_100MB.bin

Original size: 104857600 bytes (100.00 MB)

Fragments: 10 (fragment\_size=10485760 bytes)

Read time: 0.1465 s

Split time: 0.0417 s

ECC (AES-key encrypt) time: 0.0137 s

Total AES fragment encryption time (sum): 0.3757 s

Avg encrypt per fragment: 0.037566 s

Total fragment upload time (sum): 148.1377 s

Avg upload per fragment: 14.813767 s

Total uploaded bytes (fragments only): 104857920 bytes (100.00 MB)

Manifest size: 10242 bytes

End-to-end total time: 149.9806 s

Effective upload throughput (original file): 0.6668 MB/s

Effective upload throughput (uploaded bytes incl. IVs): 0.6668 MB/s

=== Performance Summary (Download & Reconstruct) ===

File: test\_100MB.bin

Original size (manifest): 104857600 bytes (100.00 MB)

Fragments: 10

AES-key ECC decrypt time: 0.0124 s

Total download time (sum): 8.6500 s

Avg download per fragment: 0.865003 s

Total AES fragment decrypt time (sum): 0.2092 s

Avg decrypt per fragment: 0.020920 s

Total downloaded bytes (fragments): 104857920 bytes (100.00 MB)

End-to-end total time (download+decrypt+merge): 93.0743 s

Effective reconstruction throughput (original file): 1.0744 MB/s

Effective reconstruction throughput (downloaded bytes): 1.0744 MB/s

**500mb**  
  
✅ File uploaded with AES-ECC hybrid encryption.

=== Performance Summary (Upload) ===

File: test\_500MB.bin

Original size: 524288000 bytes (500.00 MB)

Fragments: 50 (fragment\_size=10485760 bytes)

Read time: 0.2888 s

Split time: 0.1497 s

ECC (AES-key encrypt) time: 0.0135 s

Total AES fragment encryption time (sum): 2.2991 s

Avg encrypt per fragment: 0.045983 s

Total fragment upload time (sum): 711.2348 s

Avg upload per fragment: 14.224697 s

Total uploaded bytes (fragments only): 524289600 bytes (500.00 MB)

Manifest size: 7243 bytes

End-to-end total time: 715.7653 s

Effective upload throughput (original file): 0.6986 MB/s

Effective upload throughput (uploaded bytes incl. IVs): 0.6986 MB/s

=== Performance Summary (Download & Reconstruct) ===

File: test\_500MB.bin

Original size (manifest): 524288000 bytes (500.00 MB)

Fragments: 50

AES-key ECC decrypt time: 0.0137 s

Total download time (sum): 43.7004 s

Avg download per fragment: 0.874008 s

Total AES fragment decrypt time (sum): 1.0564 s

Avg decrypt per fragment: 0.021128 s

Total downloaded bytes (fragments): 524289600 bytes (500.00 MB)

End-to-end total time (download+decrypt+merge): 472.7407 s

Effective reconstruction throughput (original file): 1.0577 MB/s

Effective reconstruction throughput (downloaded bytes): 1.0577 MB/s

📊 **Quick Summary of Your Results (Upload Throughput)**

| **File Size** | **Avg Upload Time / Fragment** | **Effective Upload Throughput** |
| --- | --- | --- |
| 1 MB | 2.887 s | 0.241 MB/s |
| 100 MB | 14.814 s | 0.667 MB/s |
| 500 MB | 14.225 s | 0.699 MB/s |

✅ **Observation:** Upload speed improves drastically from 1 MB → 100 MB, then stabilizes at ~0.7 MB/s for 500 MB.  
**Reason:** Small files incur significant overhead from read/split/encryption. For larger files, network bandwidth is the limiting factor.

📉 **Summary of Download & Reconstruction Speed**

| **File Size** | **Avg Download Time / Fragment** | **Effective Reconstruction Throughput** |
| --- | --- | --- |
| 1 MB | 2.282 s | 0.254 MB/s |
| 100 MB | 0.865 s | 1.074 MB/s |
| 500 MB | 0.874 s | 1.058 MB/s |

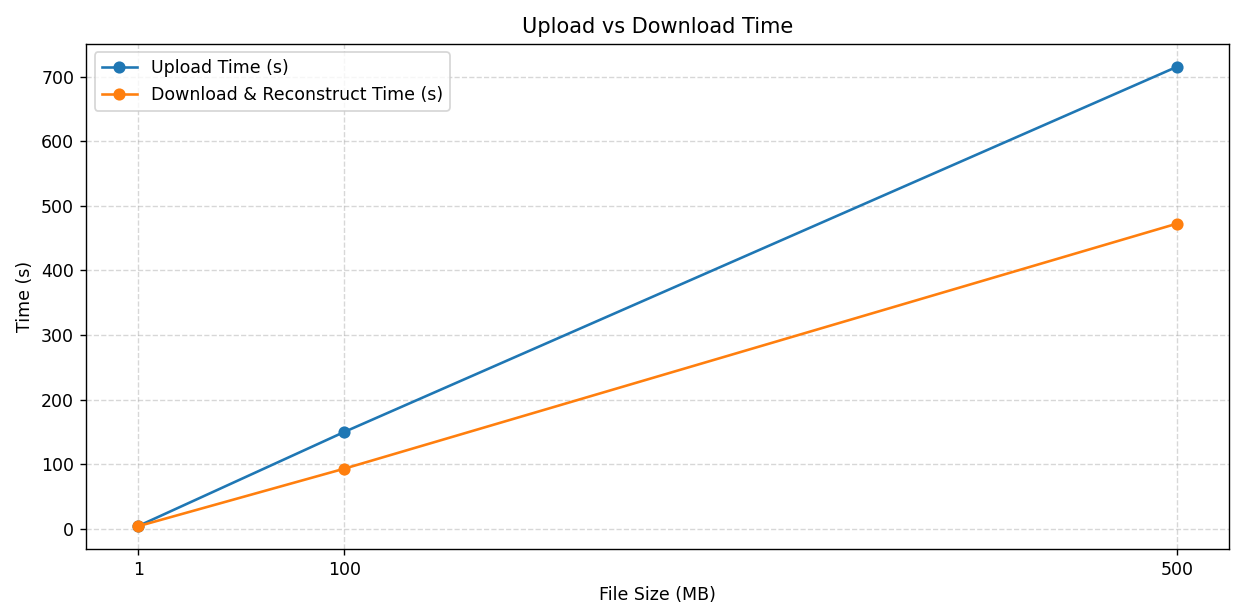
✅ **Observation:** Download is ~1.5–4× faster than upload, likely due to network asymmetry or server-side optimizations.

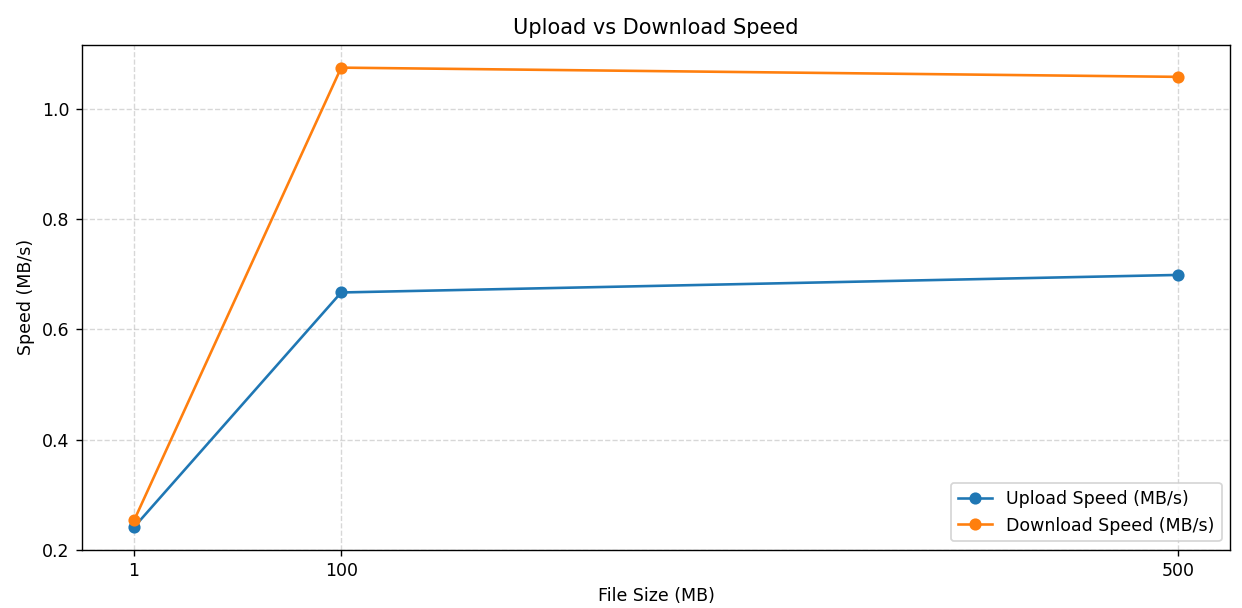
🔍 **AES-ECC Encryption Overhead**

| **File Size** | **Total AES Fragment Encryption Time** | **% of Total Upload Time** |
| --- | --- | --- |
| 1 MB | 0.0039 s | <0.1% |
| 100 MB | 0.376 s | <0.3% |
| 500 MB | 2.299 s | <0.3% |

✅ **Conclusion:** AES encryption adds negligible overhead; the network is the real bottleneck.

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





2)Fragment missing :  
  
If I remove some fragment after uploading the file and then if I try to download it shows error  
  
  
⬇️ Downloading and decrypting fragments...

✅ Fragment 349619feb2e947f99b4f17902f2ce6bb.frag done (download 5242912 bytes)

Traceback (most recent call last):

File "D:\Programs\python\HybridCloudEncryption\test\_download.py", line 125, in <module>

download\_and\_reconstruct(choice)

File "D:\Programs\python\HybridCloudEncryption\test\_download.py", line 58, in download\_and\_reconstruct

metadata, res = dbx.files\_download(dropbox\_path)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

File "D:\Programs\python\HybridCloudEncryption\.venv\Lib\site-packages\dropbox\base.py", line 1428, in files\_download

r = self.request(

^^^^^^^^^^^^^

File "D:\Programs\python\HybridCloudEncryption\.venv\Lib\site-packages\dropbox\dropbox\_client.py", line 351, in request

raise ApiError(res.request\_id,

dropbox.exceptions.ApiError: ApiError('eaa21abab59e44c182bab8bb68ec146f', DownloadError('path', LookupError('not\_found', None)))  
  
pitch:

When some fragments of the uploaded file are deleted, the download and reconstruction process fails, producing a DownloadError. This demonstrates that missing fragments prevent decryption, making the file unusable. Theoretically, this is because AES encryption operates on the complete ciphertext, and fragment loss breaks the decryption chain. Even if an attacker obtains the AES key or some fragments alone, they cannot reconstruct the original file, ensuring that both the key and all fragments are necessary for successful decryption.

3. Storage / Network Overhead

Do: Calculate:

Average increase in total bytes stored (because of IV + tag per fragment + manifest JSON).

Number of Dropbox API calls vs fragment count.

Optional: Compare single 1 GB upload vs fragmented (1 MB) upload.  
The fragmentation approach introduces additional storage and network overhead. Each fragment includes an IV, optional authentication tag, and padding, and the manifest JSON stores metadata about all fragments. As a result, the total bytes stored are slightly higher than the original file size, with the overhead increasing with the number of fragments. Network overhead also increases because each fragment requires a separate Dropbox API call; for example, uploading a 500 MB file in 50 × 10 MB fragments generates 50 API calls, compared to just one for a single large upload. This demonstrates the trade-off between security/reliability and storage/network efficiency in a fragmented encryption system.