OP-TEE Secure Storage and Chunk Management Report

Platform: Raspberry Pi 3

Storage Backend: REE File System (REE FS)

---

1. Introduction

This report explains how **objects** and **chunks** operate within the OP-TEE secure storage subsystem, how they are stored in the given setup, and what factors determine their respective sizes and limits.

2. Secure Storage Architecture

2.1 Object Concept

An **object** in OP-TEE represents a secure, persistent file stored in encrypted form within the TEE environment. Objects are physically stored by the **REE FS backend** inside the normal Linux file system (e.g., `/data/tee/`), but remain encrypted and integrity-protected.

2.2 Chunk Concept

A **chunk** is the portion of data transmitted to the TEE in a single write operation.

Concept	Description	Analogy
Object	Secure file stored in encrypted form	Locked storage box
Chunk	Small piece of data sent to fill the object	Cup of water poured into the box

3. Storage Flow

1. Create an object using `TEE_CreatePersistentObject()`.
2. Write data in successive chunks using `TEE_WriteObjectData()`.
3. Each chunk (64 KB) is encrypted and written to REE FS.
4. Close the object using `TEE_CloseObject()`.

4. Configuration Parameters Affecting Chunk and Object Size

Parameter	Description	Observed Value	Impact
CFG_REE_FS_TA_BUFFER_SIZE	Buffer size for REE FS read/write operations	64 KB	Defines maximum per-chunk transfer size
CFG_CORE_HEAP_SIZE	Total heap available to TEE core	64KB	Temporary memory available for buffers
CFG_TEE_RAM_VA_SIZE	Virtual address space of TEE OS	7 MB	Defines total usable memory
CFG_SHMEM_SIZE	Shared memory between REE and TEE	4 MB	Limit for data exchange per RPC
Backend Type	Storage mechanism	REE FS	Object stored on Linux FS (encrypted)

5. Object and Chunk Size Analysis

5.1 Chunk Size

- Determined by `CFG_REE_FS_TA_BUFFER_SIZE`.
- On this setup: **64 KB**.

5.2 Object Size

- No strict limit with REE FS backend.
- Data streamed chunk-by-chunk to disk.
- Practical limit depends on SD card capacity.

6. Example Operation

Writing a 1 MB file in 64 KB chunks:

```
size_t chunk = 64 * 1024;
for (size_t i = 0; i < file_size; i += chunk)
    TEE_WriteObjectData(object, data + i, chunk);
```

Step	Action	Data Processed
1	Create object	—
2	Write chunk #1	64 KB
...	...	...
16	Write final chunk	64 KB
17	Close object	File complete (1 MB)

7. Purpose of Storing Files Inside TEE

- Security: Data is encrypted, hidden from Linux.
- Integrity: Cannot be tampered with.
- Persistence: Survives reboot.
- Confidentiality: Only owner TA can access.
- Device binding (if RPMB used): Bound to hardware.

8. Summary of Findings

Aspect	Finding
Object Type	Secure, encrypted file stored in REE FS
Chunk Size (maximum)	64 KB
TEE Virtual Memory	7 MB
Shared Memory	4 MB
Per-object Limit (REE FS)	Practically unlimited (depends on SD card)
Write Mechanism	Sequential (streaming) using chunks
Storage Location	/data/tee/ (SD card)
Encryption & Integrity	Managed by OP-TEE core

9. Conclusion

In this Raspberry Pi 3 configuration, OP-TEE uses **chunk-based streaming writes** to store data securely. Each 64 KB chunk is encrypted and saved to the REE filesystem, forming one large object. Memory limits apply per chunk, while total size is limited only by disk capacity. This enables secure, persistent, and scalable storage in the Trusted Execution Environment.