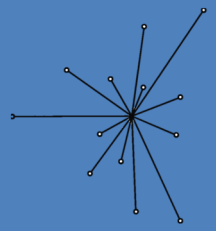


HYU IoT and ASU KIM Lab

Secure and Lightweight Access Control for Highly Decentralized and Distributed File Systems

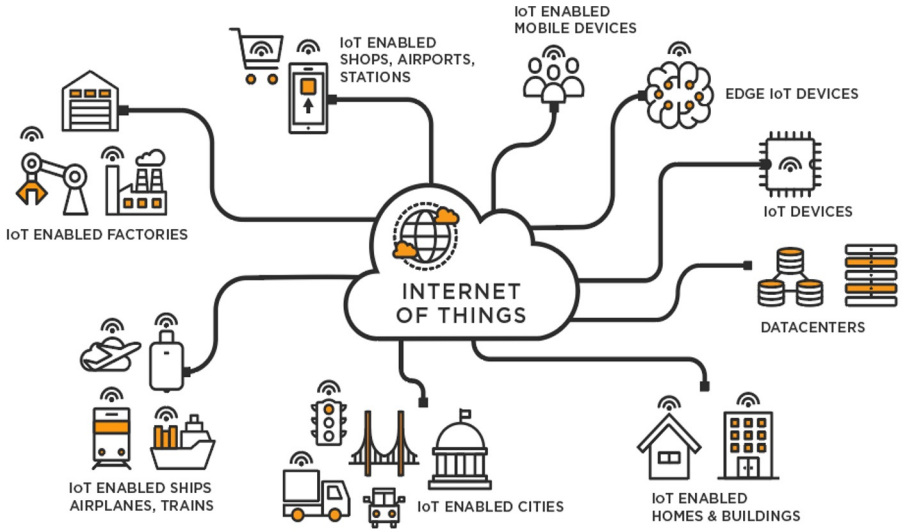
Yeongbin Jo[1], Yunsang Cho[1], and Hokeun Kim[1,2]

[1]**Hanyang University and** [2]**Arizona State University**
HYU IoT Lab and ASU KIM lab

HYU IoT Lab: https://hyu-iot.github.io/
ASU Kim Lab: https://labs.engineering.asu.edu/kim/

Overview – IoT & Cloud

- Internet of Things (IoT)



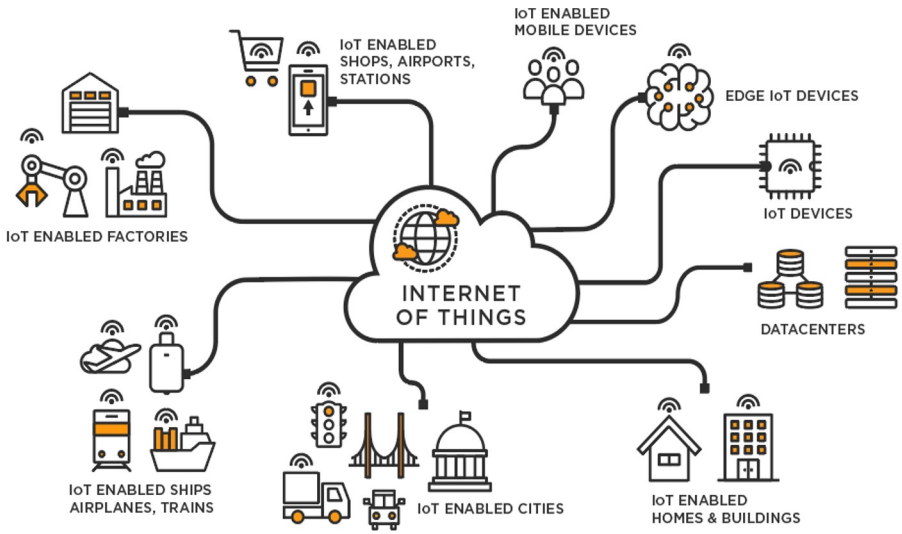
- Cloud

- **Proposed approach – Securing decentralized, distributed file system *using SST***

- Decentralized distributed file system: File system without central server
- SST: Secure Swarm Toolkit which provides secure key for access control

Motivation

Challenges in IoT & Cloud

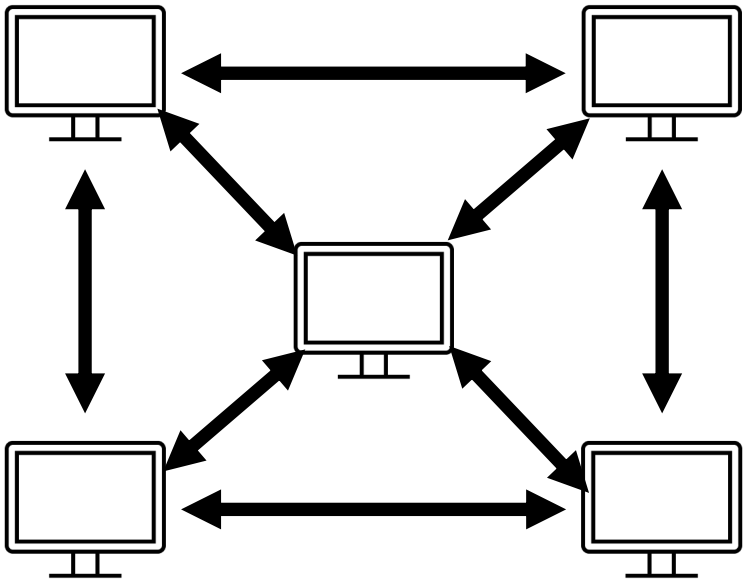


- **Dependence on Internet connection**
- **Costs for using cloud services**

Alternative: Decentralized distributed file system

Motivation

Decentralized, distributed file system



- **Dependence on Internet connection**

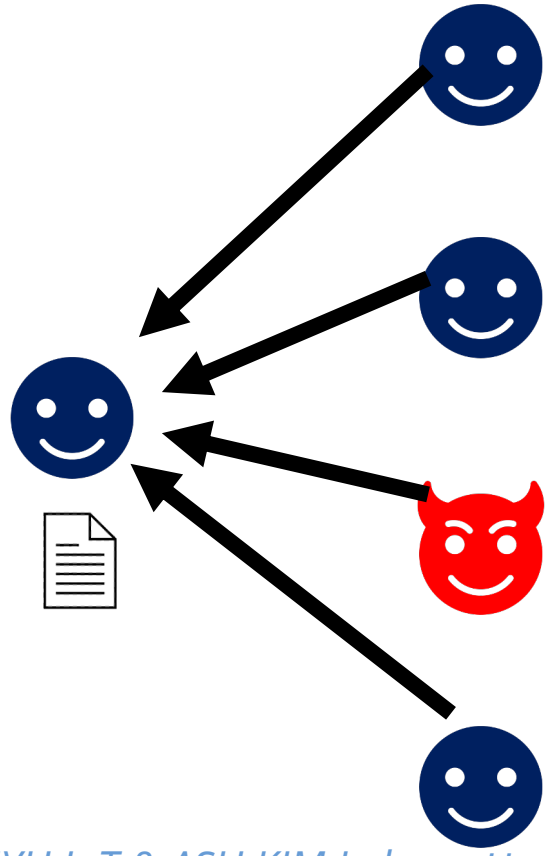
-> Can work even when Internet connection is unstable

- **Costs for using cloud services**

-> Much lower cost (IPFS)

Motivation

Challenges in free decentralized, distributed file system (e.g., IPFS, BitTorrent)



1. Anyone can download the file if downloader knows information of the file

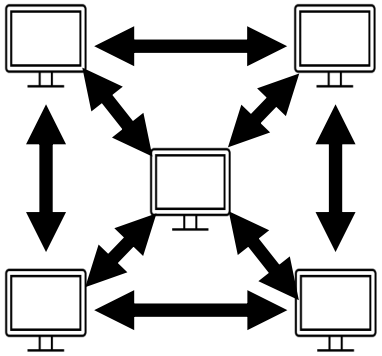
2. Provider may be difficult to provide the file to specific person or group

-> We propose secure and lightweight mechanism for access control to decentralized, distributed file system

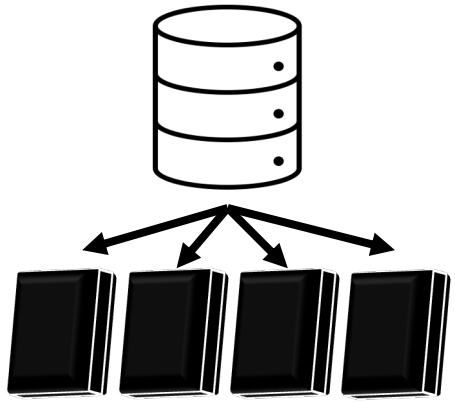
Background

Decentralized, distributed file system

- **Decentralization – there is no central server. File can be downloaded from nodes which have the file.**

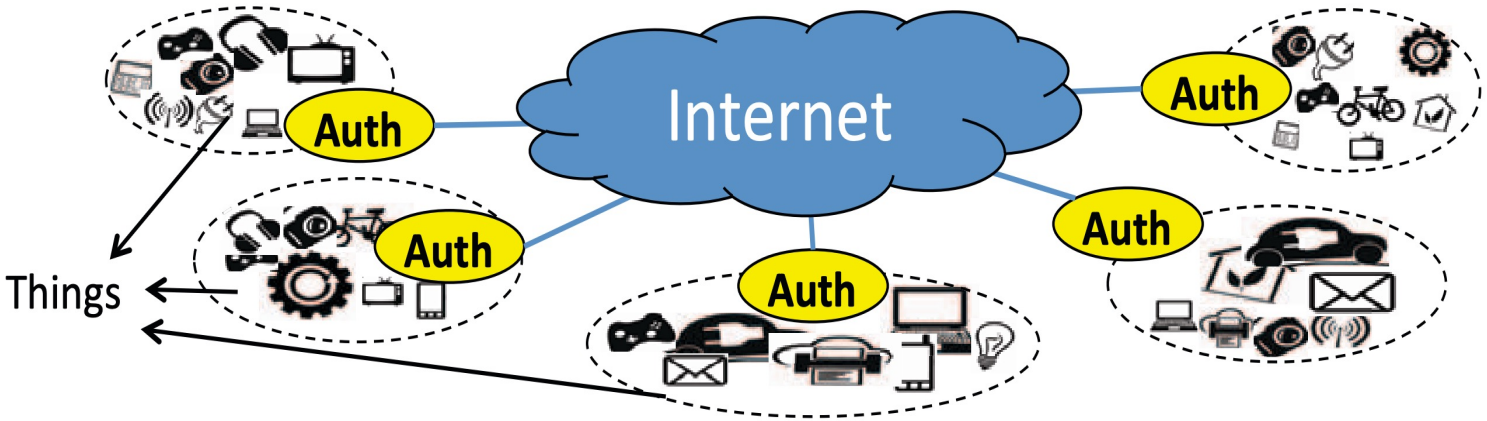


- **Distribution - Files are divided into smaller pieces or blocks.**



Background

Secure Swarm Toolkit (SST)

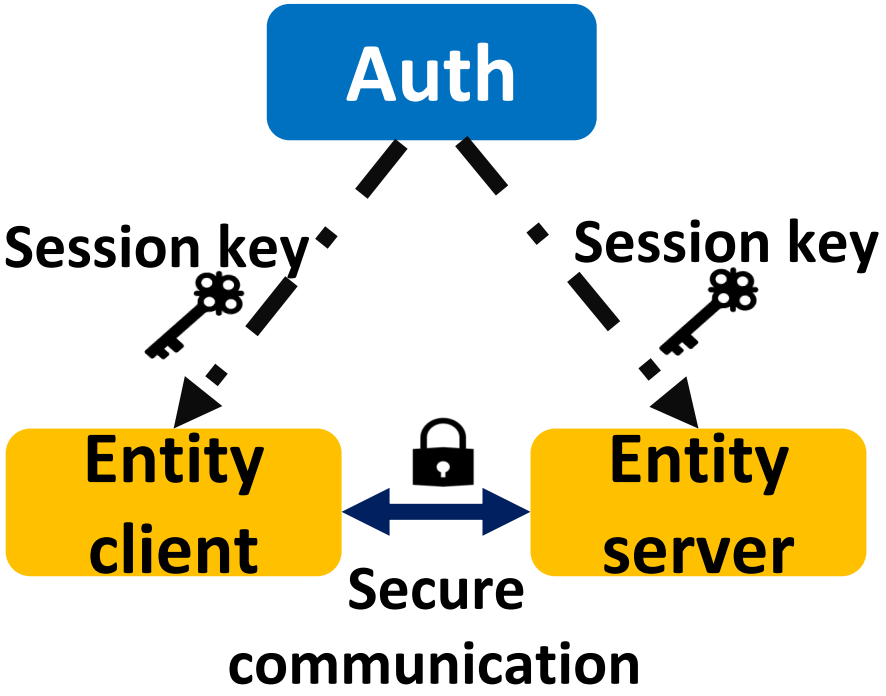


(Figure from Kim et al. IoTDI17. 'A toolkit for construction of authorization service infrastructure for the internet of things')

SST is appropriate to distributed system!!

Background

Secure Swarm Toolkit (SST)



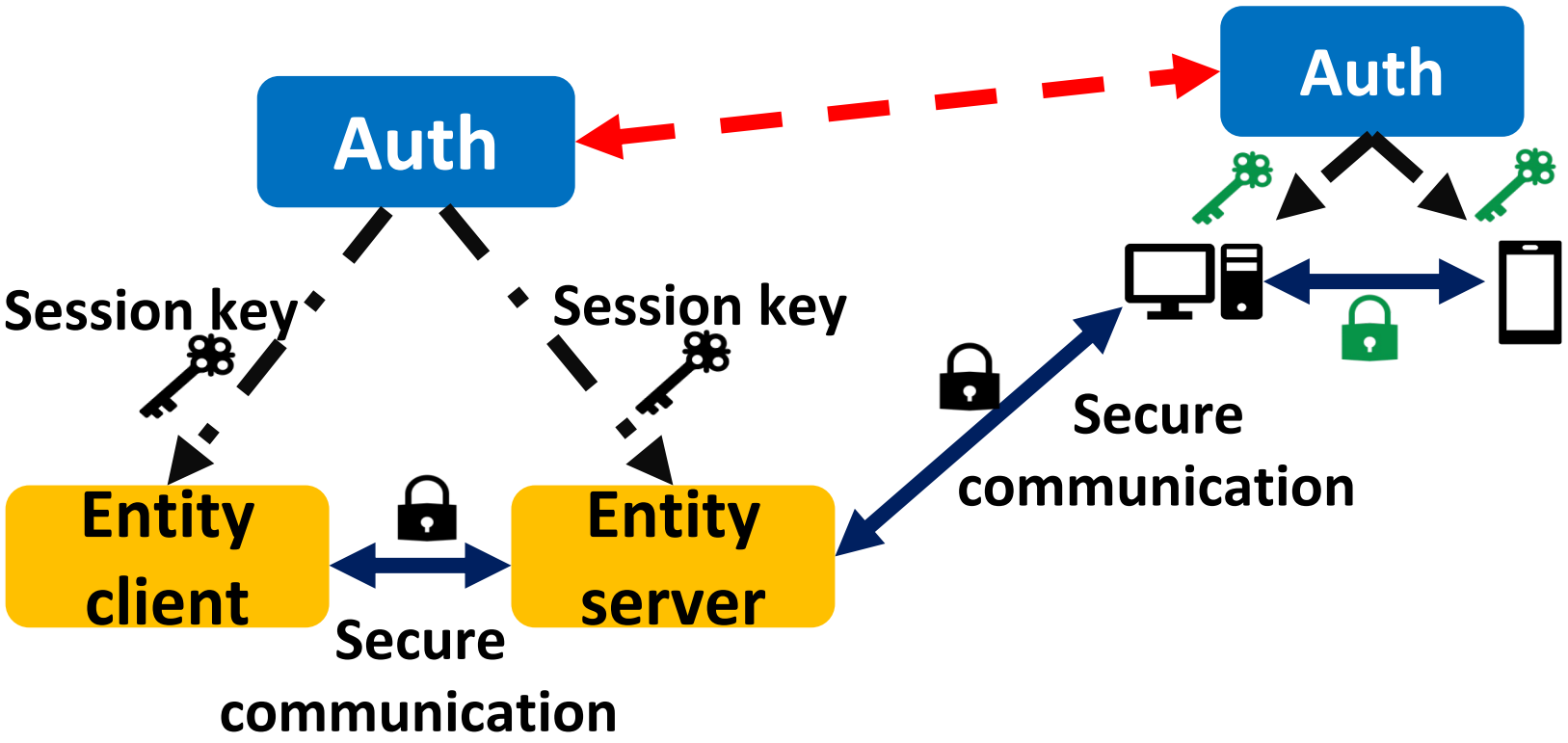
Auth: responsible for authenticating and authorizing registered entities

Entity: any device connected to the network in the IoT to be authenticated and authorized

Session key: a symmetric key used to protect a single session of communication

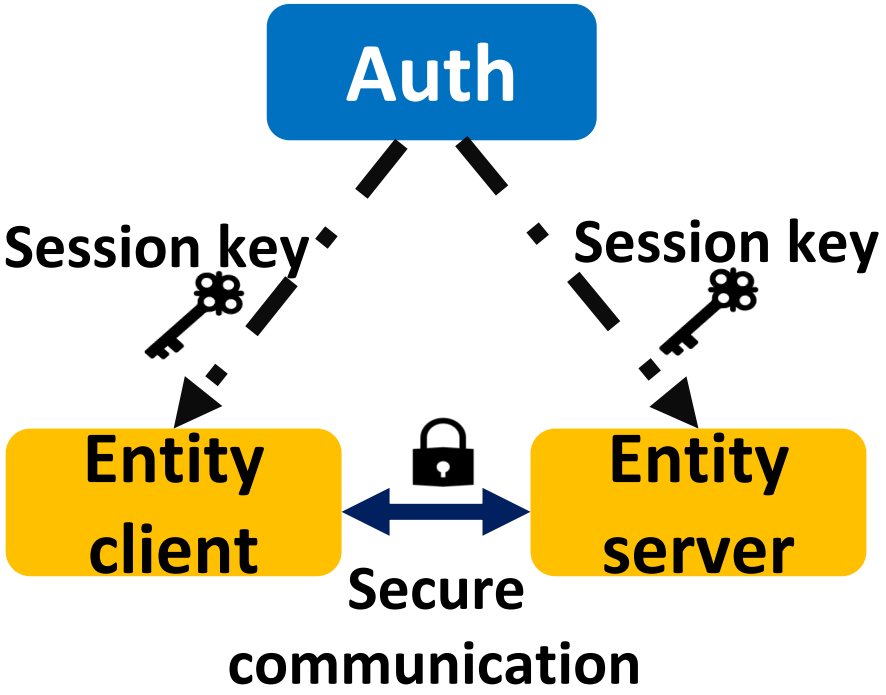
Background

Secure Swarm Toolkit (SST)



Background

Secure Swarm Toolkit (SST)



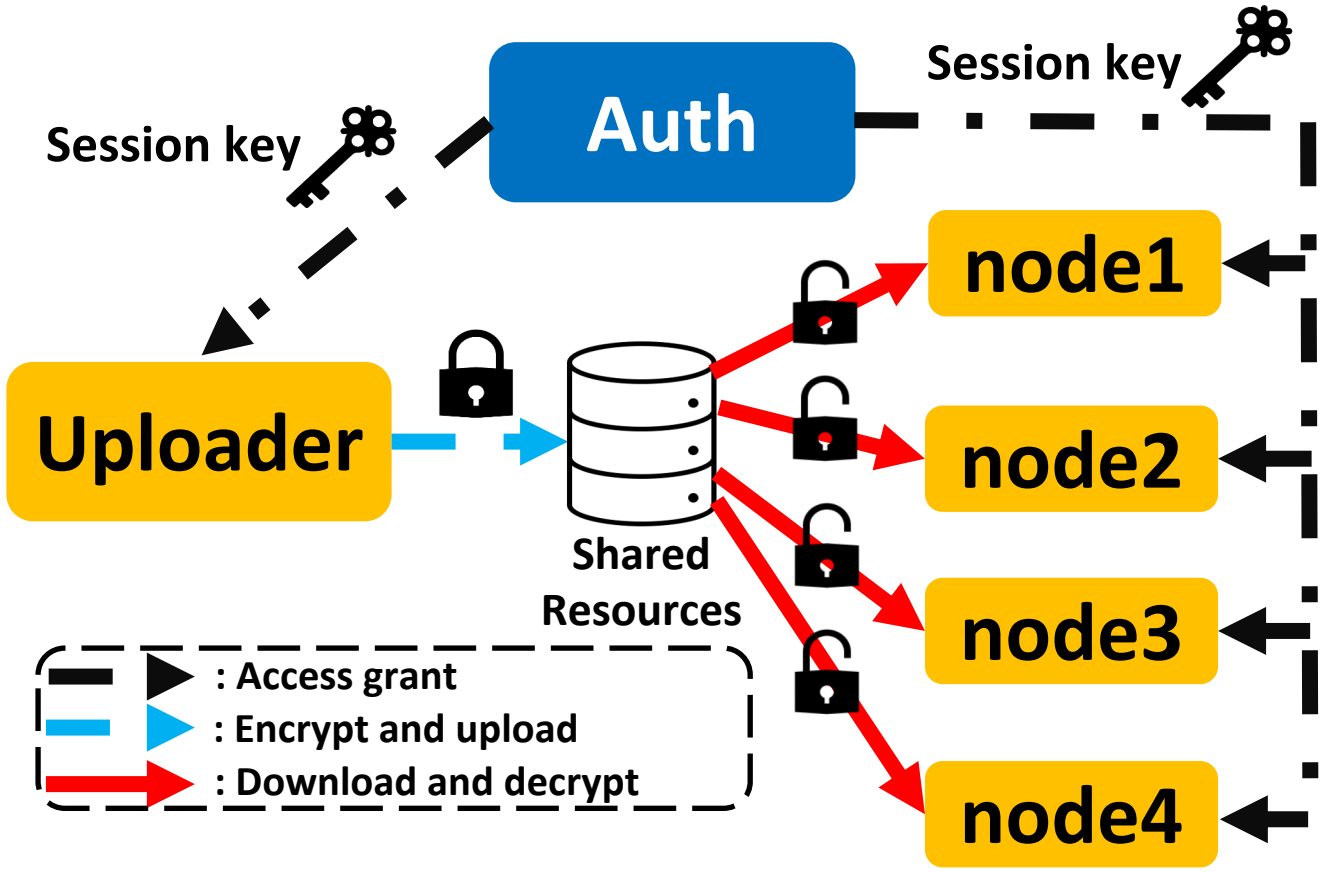
1. Heterogeneity
-> Support various security configurations

2. Open environment
-> Revoke credentials of compromised entities

3. Scalability
-> Use multiple Auths

Approach

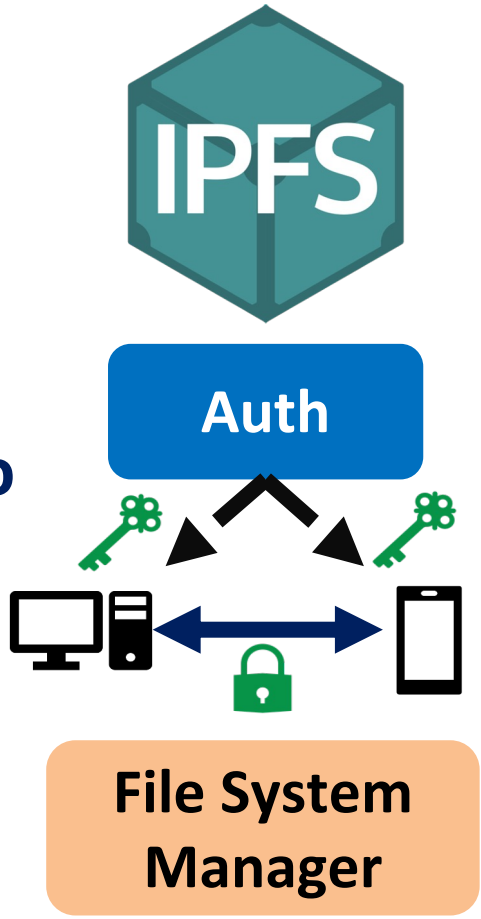
Decentralized, distributed file system with SST



Approach

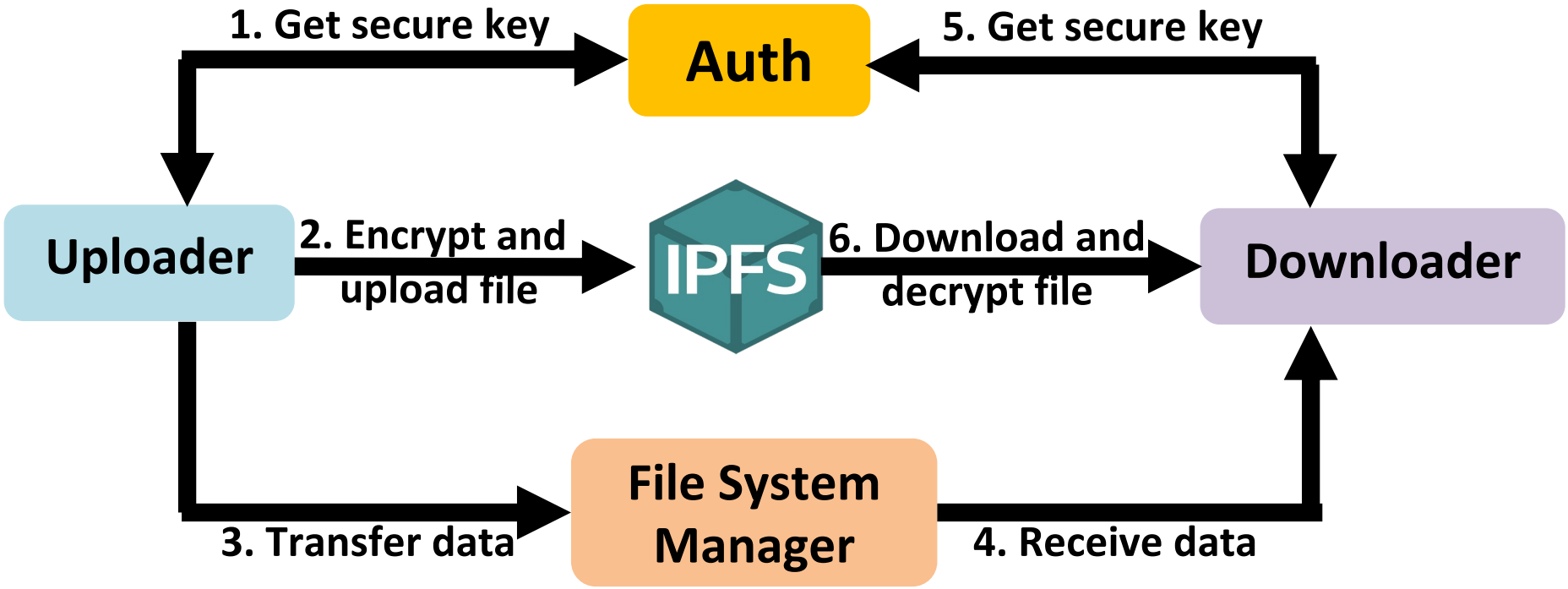
Decentralized, distributed file system with SST

- **We use Inter-planetary File System (IPFS) which is a decentralized, distributed file system.**
- **We use SST which provides secure key to encrypt and decrypt file.**
- **We design and develop file system manager software for meta information (e.g., key id, file hash)**



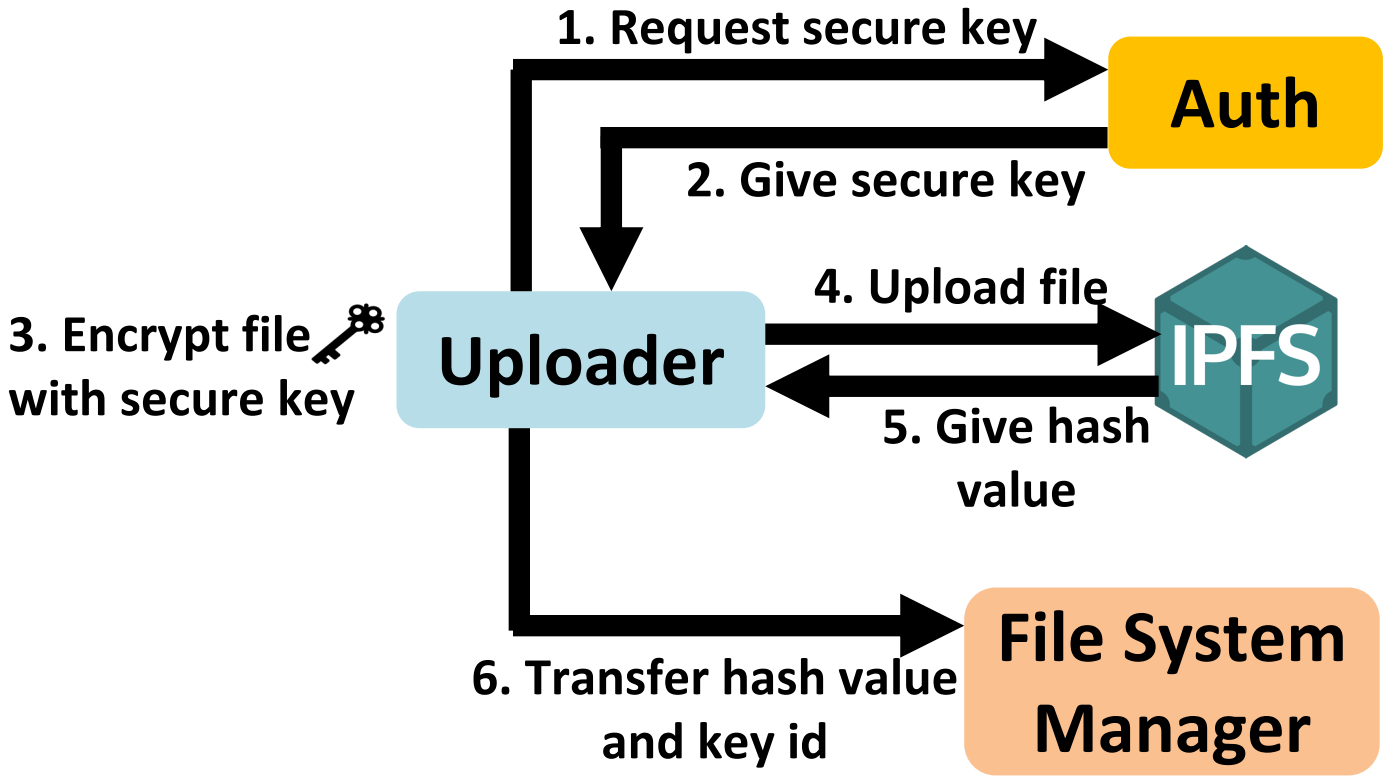
Approach - IPFS with SST

Operation



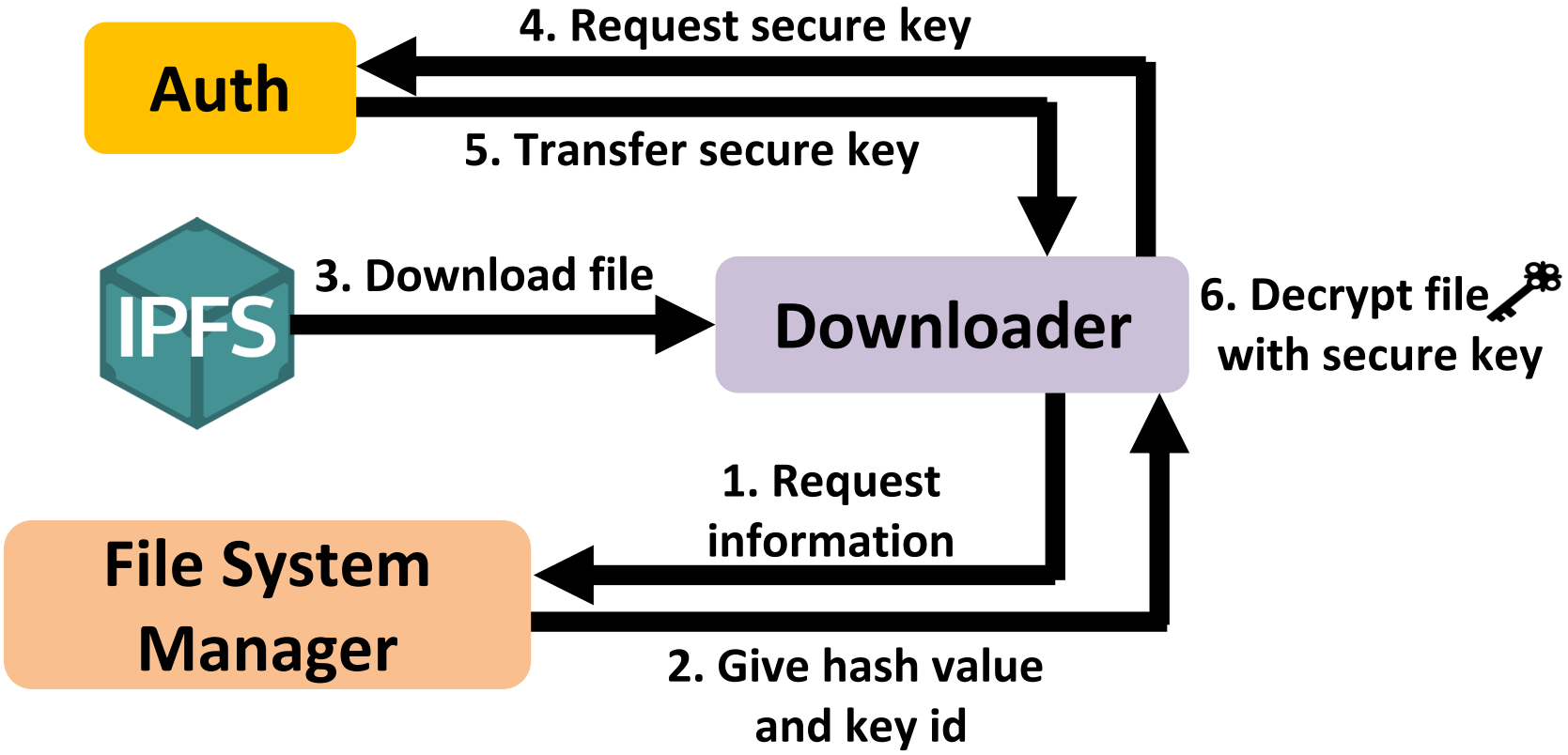
Approach - IPFS with SST

File upload operation



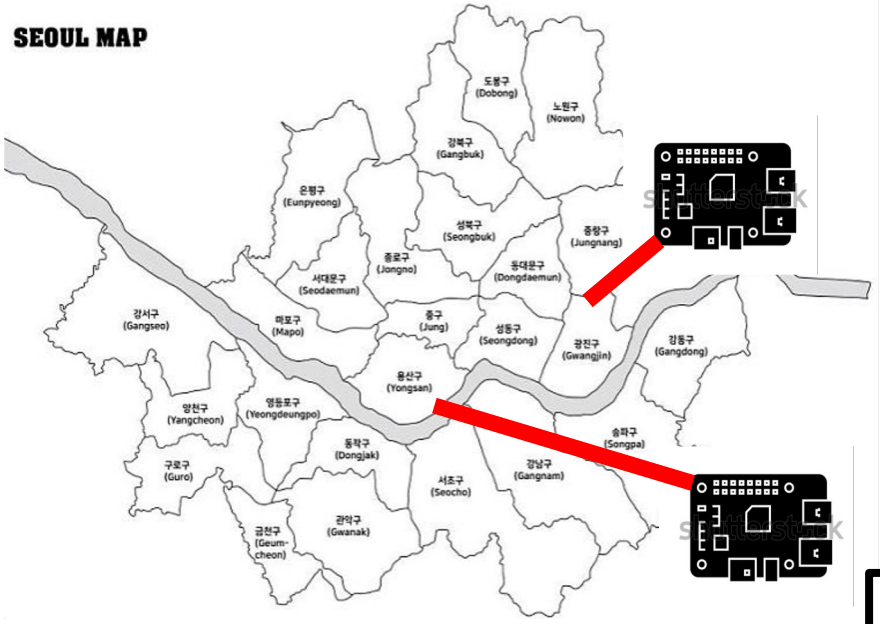
Approach - IPFS with SST

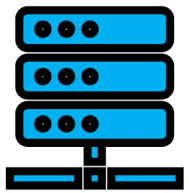
File download operation



Evaluation

Experimental Setup





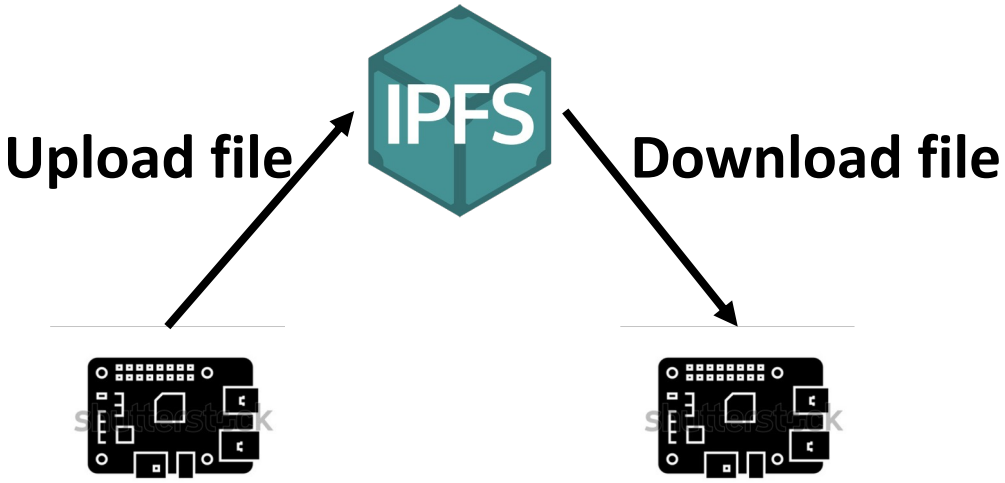
Auth

File System Manager

- Entity - RaspberryPi4
- Auth, File System Manager

Evaluation – scenario1

Compare up/download total execution time with and without SST

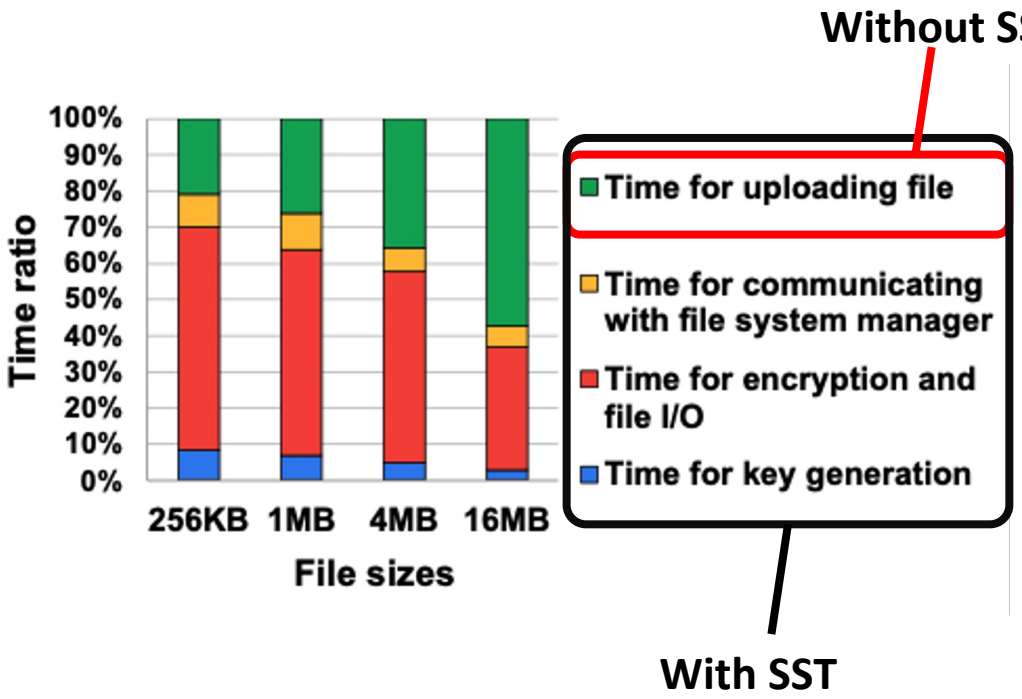


- The experiment is conducted at 256 KB, 1 MB, 4 MB, and 16 MB.

- Estimate execution time at each step

Evaluation – scenario1

Result for uploading files



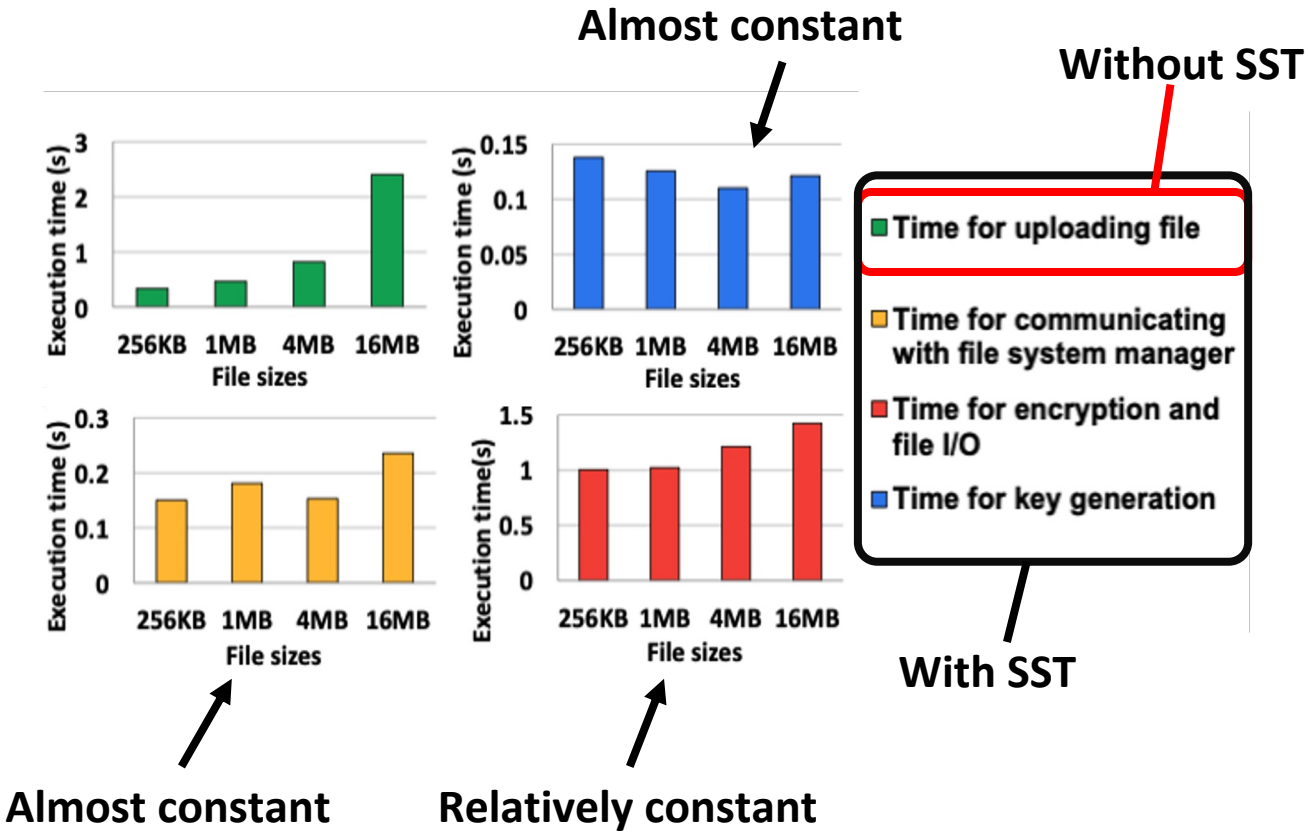
- Uploading time is dominant as the file grows in size.

- Time for encryption and file I/O is high due to operating several processes.

- Execution overhead of SST is 42% at 16 MB.

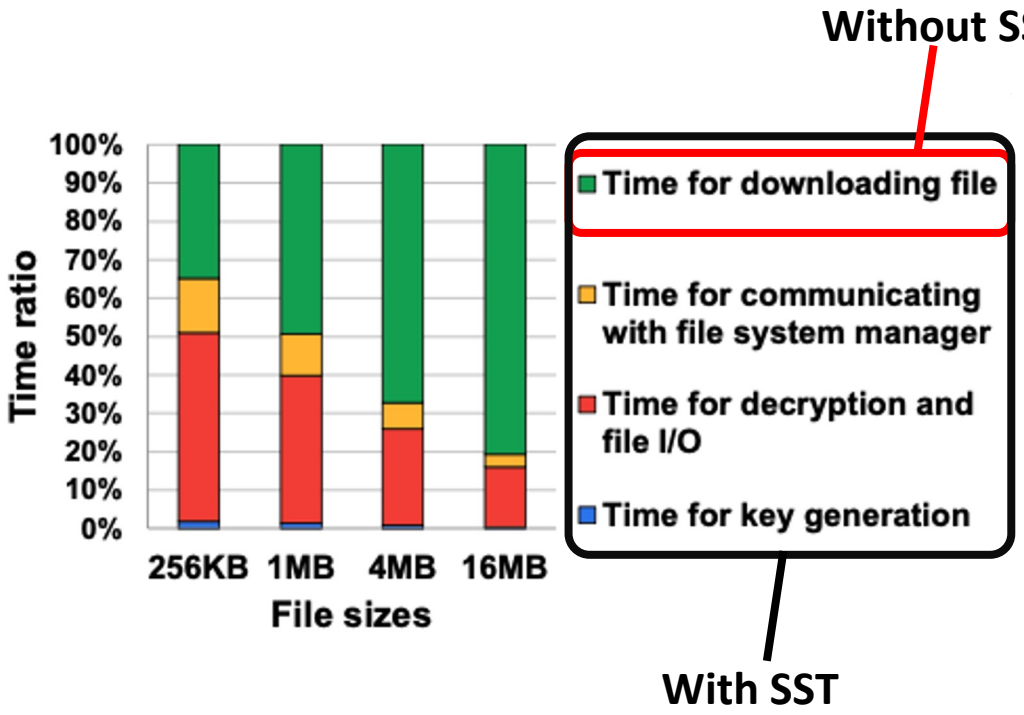
Evaluation - Scenario 1

Result for uploading files



Evaluation - Scenario 1

Result for downloading files



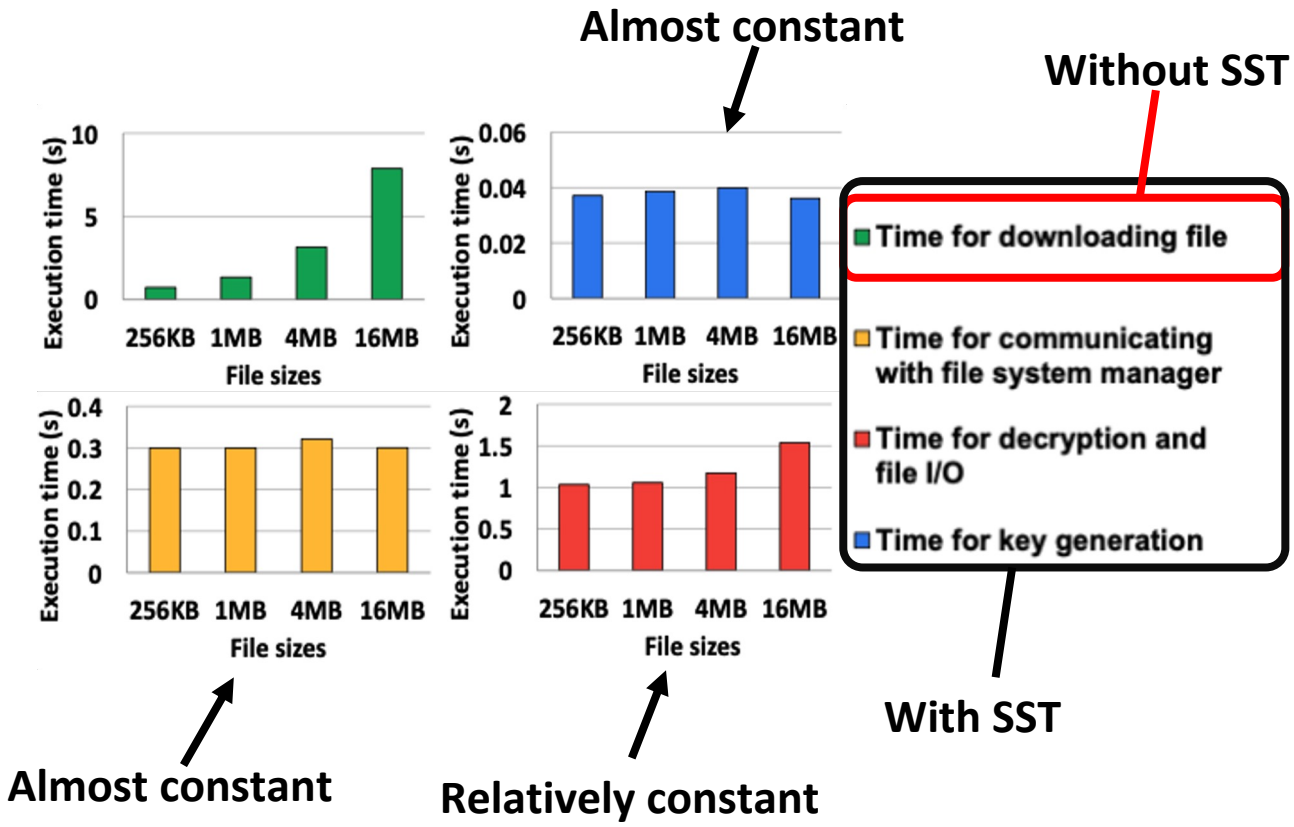
- Downloading time is dominant as the file grows in size.

- Time for encryption and file I/O is high due to operating several processes.

- Execution overhead of SST is 19% at 16 MB.

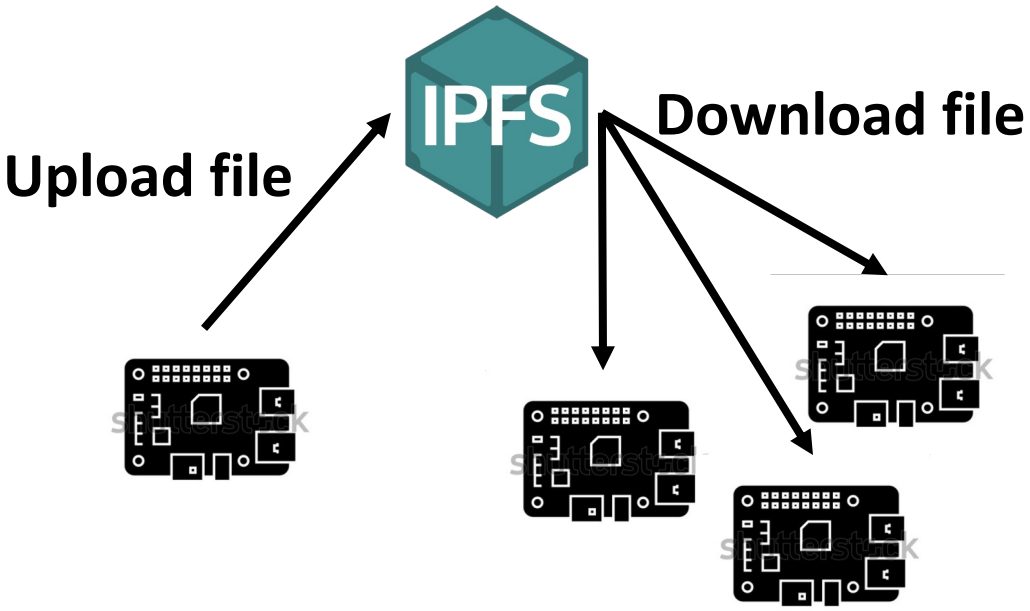
Evaluation - Scenario 1

Result for downloading files



Evaluation – Scenario 2

Compare multi-download total execution time with SST

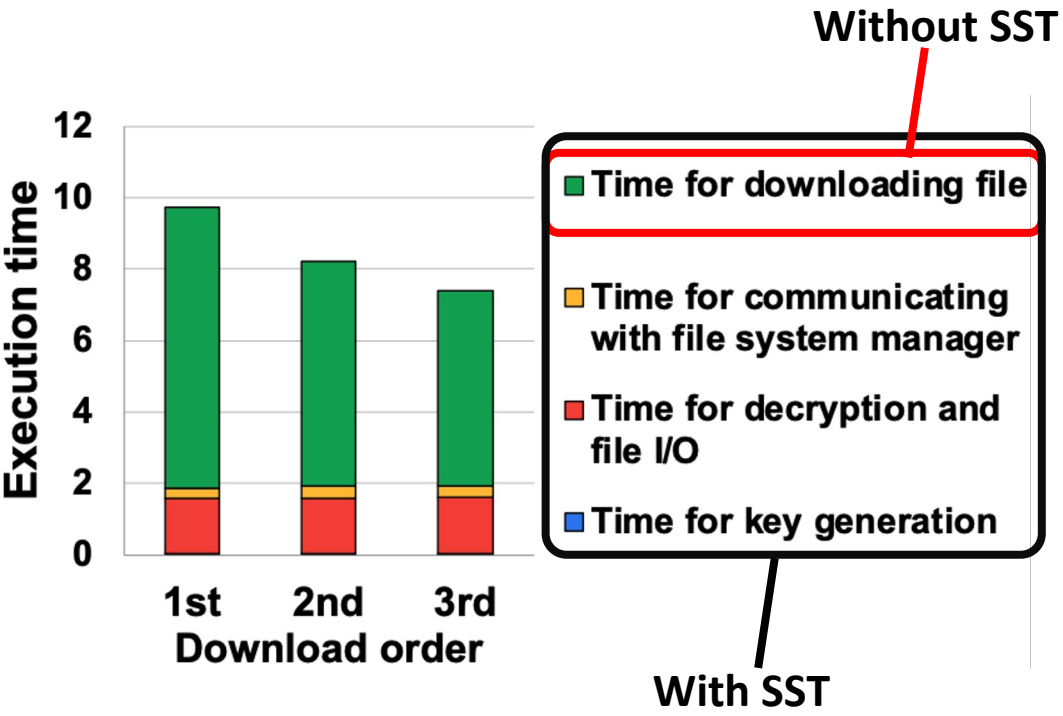


- The experiment is conducted at 16 MB.

- Estimate total execution time according to download order.

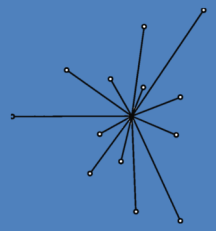
Evaluation – Scenario 2

Result for multi-downloaders



- Download takes less time when there are more sharers

- Other time is not affected.



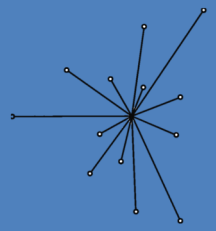
Related Work

- **Storj**

  - Provides access control using Macaroons(bearer token + caveat + HMAC)

- **Smart contract**

  - Provides access control using smart contract to verify permissions of the nodes



Conclusion

- **Security solution with minimal cost tailored to decentralized, distributed file systems**
  - Low additional execution time
  - Low additional network usage

- **Future work**
  - Design more robust mechanism for managing the resources of the distributed files.
  - Carry out more in-depth evaluation of the security and efficiency of our approach at greater scale.

Thank you