Instant Auditing of Cloud Storage Access without Accumulating Attestations

Advicer : Gwan-Hwan Hwang
Student : Wei-Chih Chien

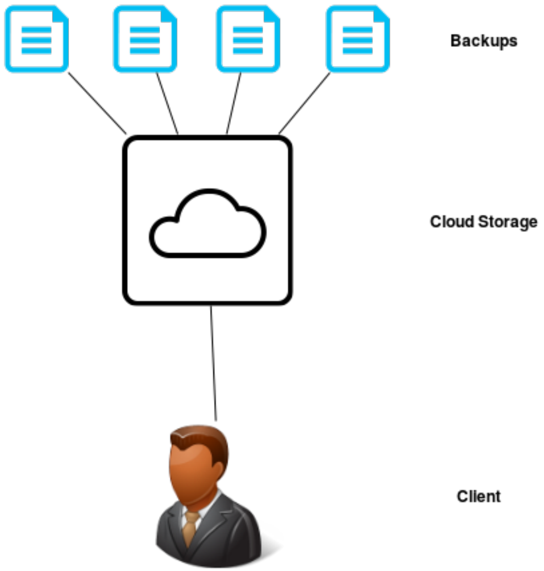
NTNU CSIE CCLAB

2015.09.02

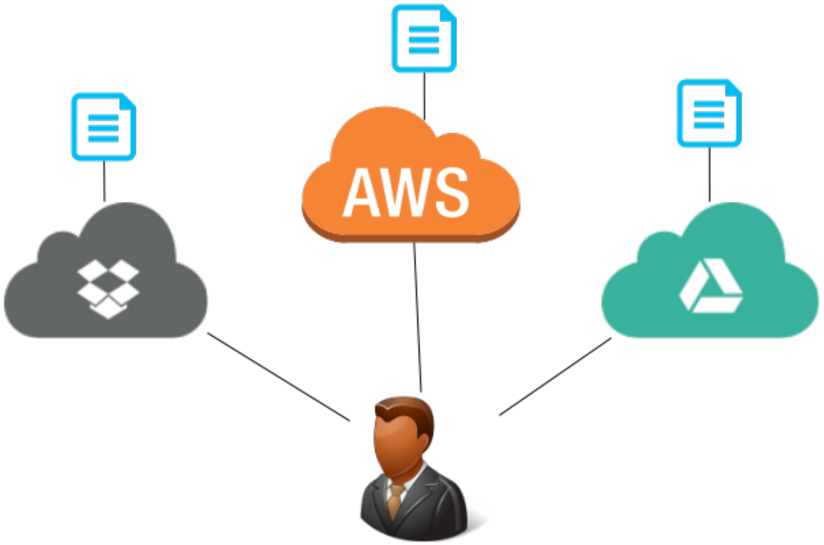
Outline

Outline

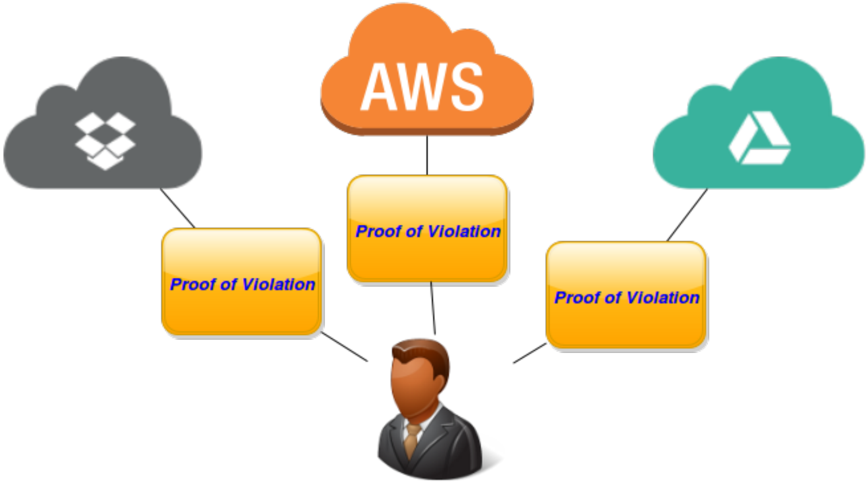
Scenario



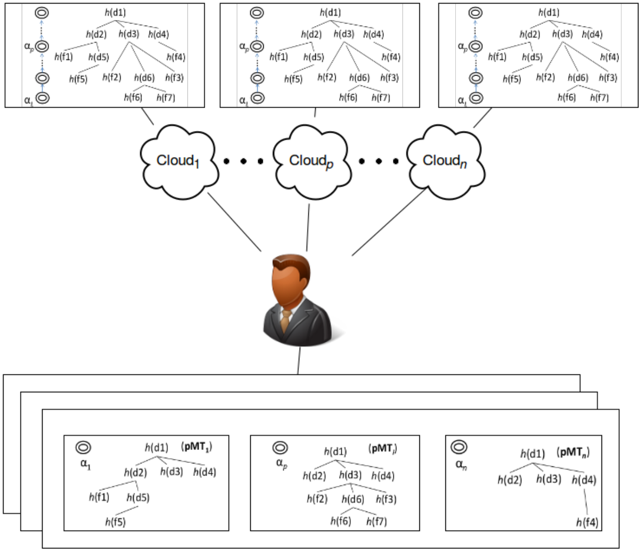
Scenario - Backup



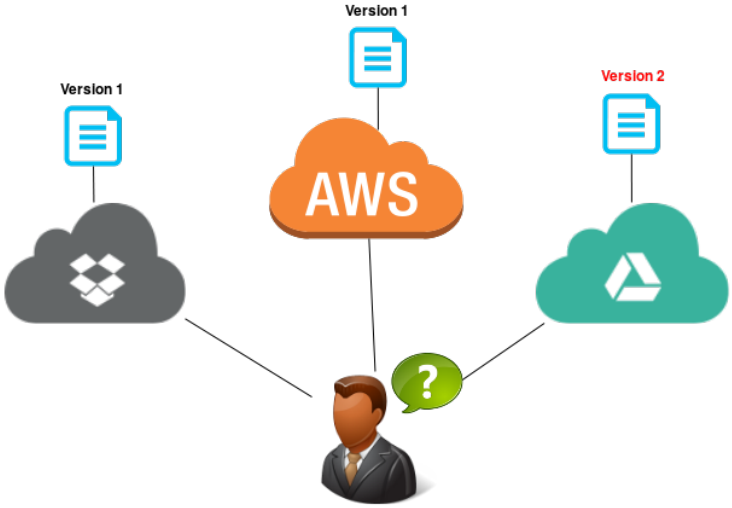
Scenario - POV



Scenario - Problem I : Too Many Attestations

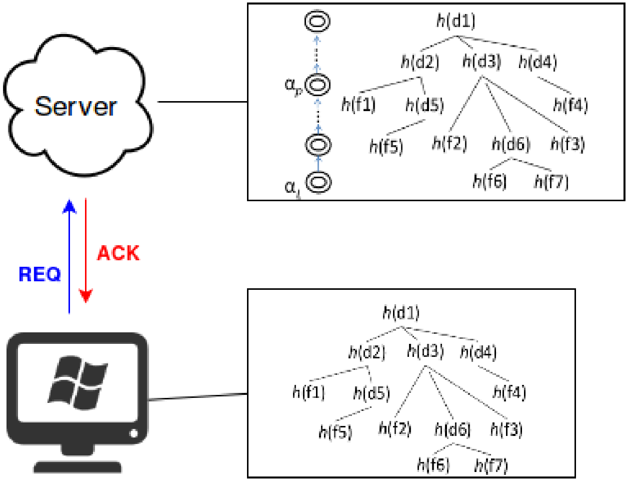


Scenario - Problem II : Version Control is Difficult



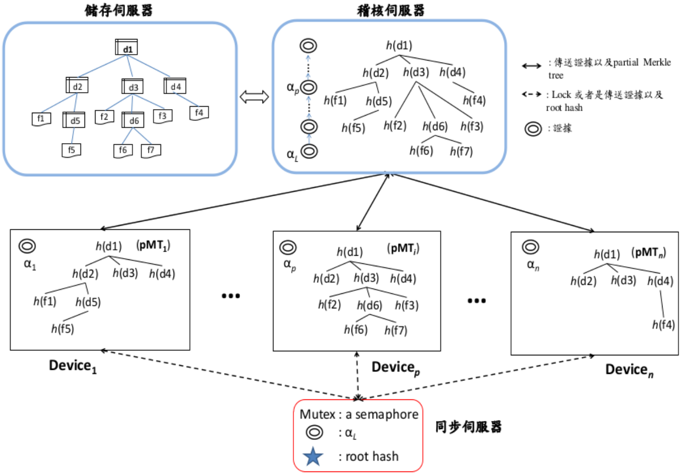
Outline

Single Client and Service Provider



Multiple Clients

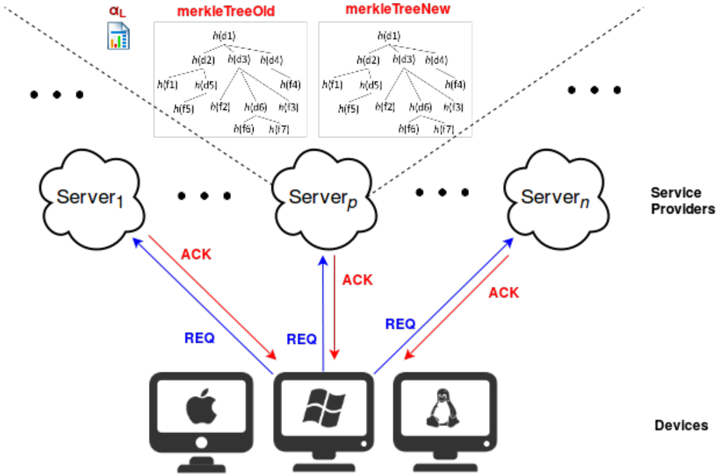
(*Instant Auditing of Cloud Storage Access by Cache Partial Merkle tree*)



Worst-case：很久沒更新而累積了大量的files update動作

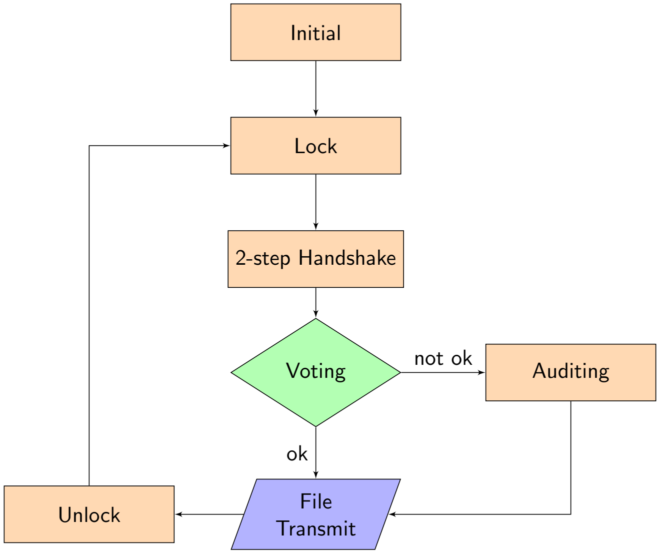
Reduce Device's Storage Usage

Assumption: 同時有k個server上同一file出問題的機率 ≈ 0



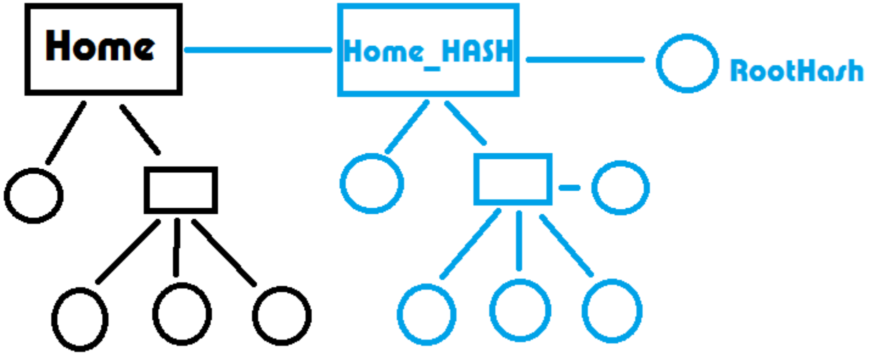
Outline

Flowchart


Initial
Lock
2-step Handshake
Voting
not ok
Auditing
ok
File Transmit
Unlock

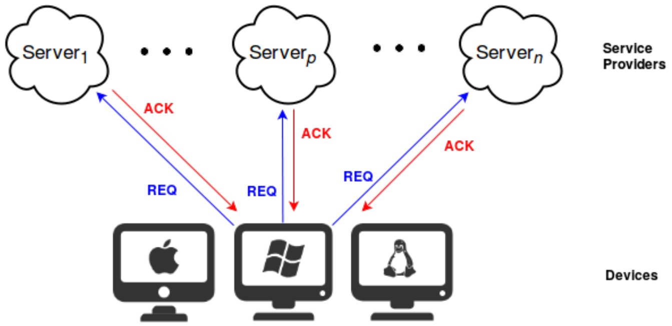
Initial

File → Merkle Tree



READ

I. 2-step Handshake

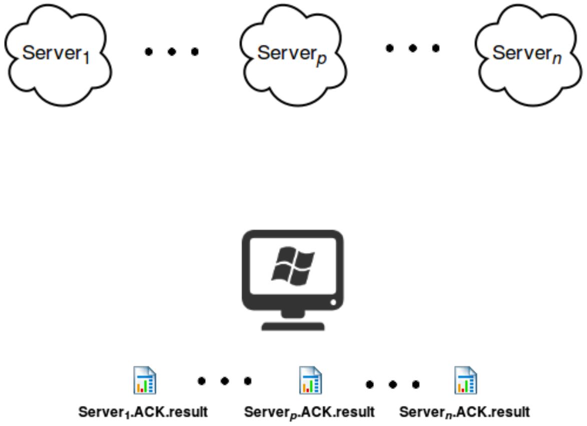


REQ = (op, $[\text{op}]_{\text{pri(D)}}$)
op.type = DOWNLOAD
op.path = filepath
op.msg = " "

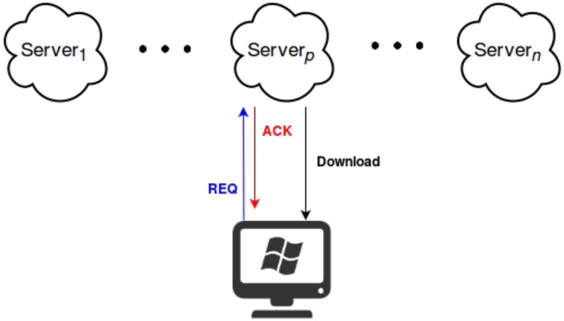
ACK = (result = merkleTreeNew.roothash,
REQ,
$[\text{result, REQ}]_{\text{pri(S)}}$)

READ

II. Voting


Server_1
Server_p
Server_n
Server_1.ACK.result
Server_p.ACK.result
Server_n.ACK.result

READ
III. Download

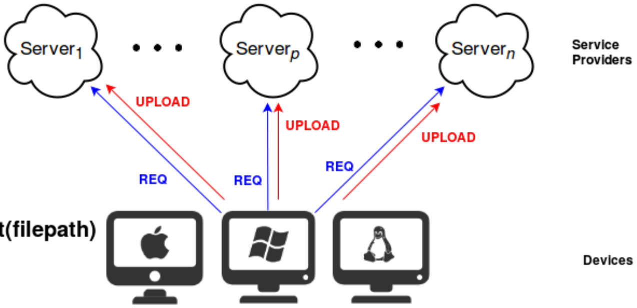


REQ = (op', $[\text{op'}]_{\text{pri(D)}}$)
op'.type = DOWNLOAD
op'.path = op.getPath()
op'.msg = ACK.result

ACK = (result = merkleTreeNew.roothash,
REQ,
$[\text{result, REQ}]_{\text{pri(S)}}$)

WRITE

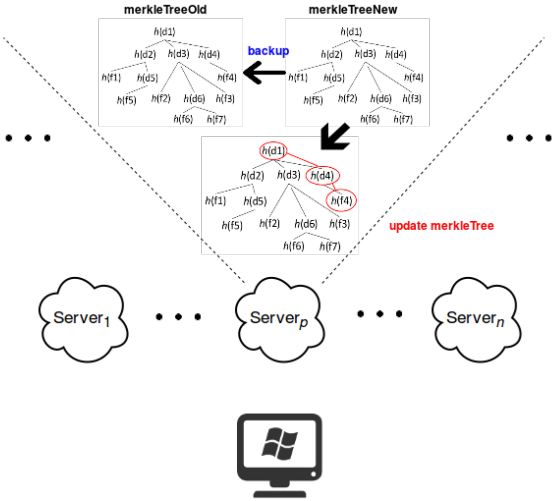
I. Upload



REQ = (op, $[op]_{pri(D)}$)
op.type = UPLOAD
op.path = filepath
op.msg = Utils.digest(filepath)

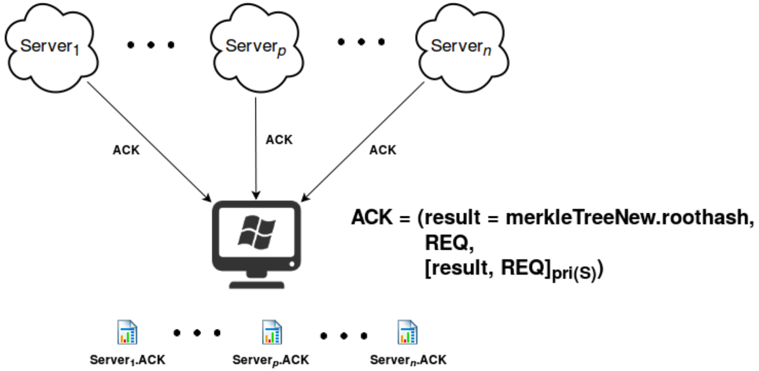
WRITE

II. Update Merkle Tree



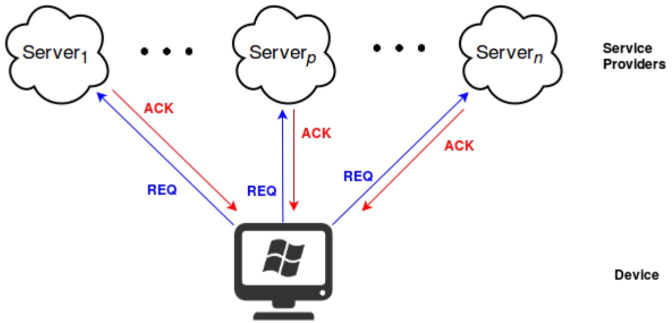
WRITE

III. Voting



AUDIT

I. Download and Check LastReq

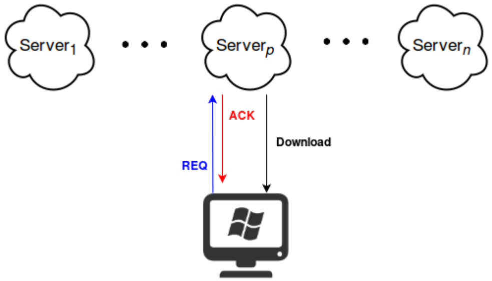


REQ = (op, $[op]_{pri(D)}$)
op.type = AUDIT
op.path = " "
op.msg = " "

ACK = (result = merkleTreeNew.roothash,
REQ = lastReq,
$[result, REQ]_{pri(S)}$)

AUDIT

II. Download Roothash or Merkle Tree



```
REQ = (op, $[op]_{pri(D)}$)
op.type = AUDIT
op.path = attestationPath
op.msg = " "
```

```
ACK = (result = Utils.digest(file),
       REQ,
       $[result, REQ]_{pri(S)}$)
if lastReq.op == DOWNLOAD:
    file = merkleTreeOld.roothash
if lastReq.op == UPLOAD:
    file = serialize(merkleTreeOld)
```

AUDIT

III. Auditing

```
AUDIT(lastAck)
    op ← lastAck.req.op
    if op.type = DOWNLOAD
        success ← roothash.equals(lastAck.result)
    if op.type = UPLOAD
        merkleTreeOld.update(op.msg)
        success ← roothash.equals(lastAck.result)
    return success
```

Listing 1 : Audit algorithm

Outline

Implement Steps

1. Hash Handle : create, update, delete
2. Operation Handle : read, write, audit
3. File Transmit : send, receive
4. Merkle Tree Transmit : serialize
5. *Instant Auditing of Cloud Storage Access by Cache Partial Merkle tree*
6. Run on Real Cloud Environment (VM)

Outline

Create Merkle Tree

Account A	666 MB	42 files	6 directories
Account B	34 MB	54192 files	188 directories
Account C	6.54 GB	58484 files	1718 directories
Account D	20.6 GB	175389 files	5154 directories

Table : TIMES REQUIRED TO GENERATE THE ROOT HASH FROM NOT-HASHED FILES (IN SECONDS)

Account	**Senior**	**My**	**MerkleTree Size**
A	*3.404*	*3.645*	3.74 KB
B	*16.618*	*7.669*	3.77 MB
C	*229.351*	*242.198*	4.30 MB
D		*815.408*	12.9 MB

Operation Processing Time

Table : DOWNLOAD TIME (ms)

Account	100 times	Audit*
A	*4635*	*34 + 0*
B	*4660*	*33 + 0*
C	*5429*	*31 + 0*
D	*5554*	*31 + 0*

Table : UPLOAD TIME (ms)

Account	100 times	Audit*
A	*4322*	*41 + 7*
B	*5643*	*421 + 997*
C	*9236*	*421 + 2621*
D	*11466*	*1263 + 8085*

* download attestations time + audit time