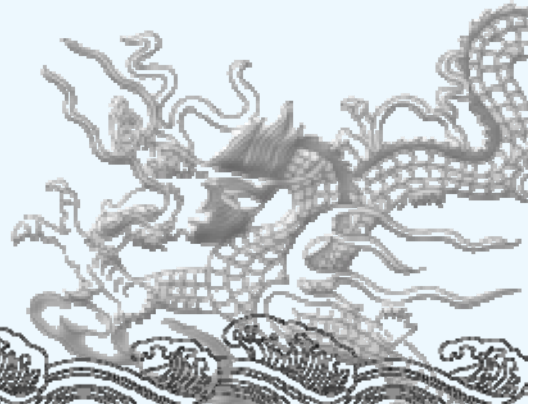
Enabling Efficient Batch Verification on Data Integrity for Cloud

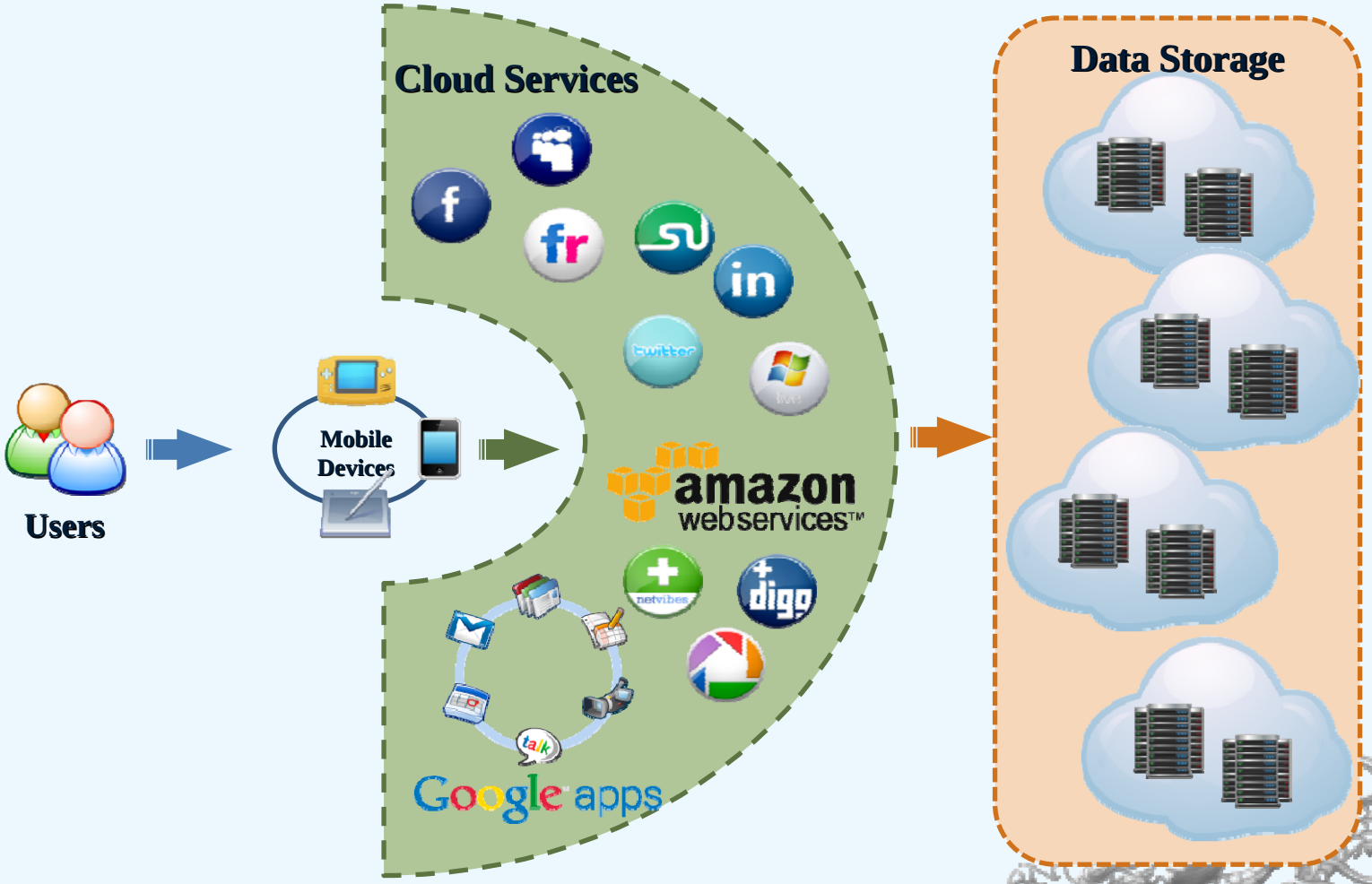
Chin-Laung Lei
Department of Electrical Engineering
National Taiwan University

Outline

- Introduction
- System model
- Protocol
- Experiments and performance analysis
- Conclusion



Cloud Computing



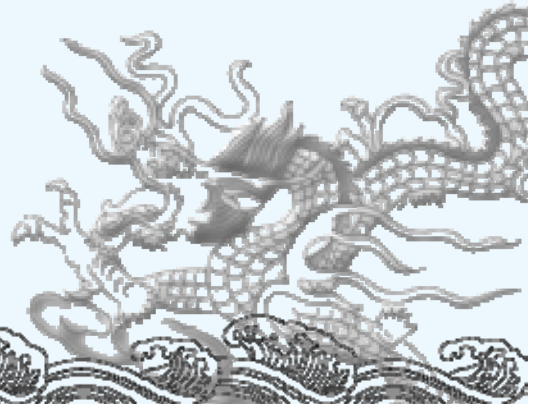
Motivations

- Online sharing and co-working become a trend. ex:

- We want to know everyone's modification are intact and to make modifications undeniable.
- We wish our protocol can be applied to most of existing PDP schemes based on public-key cryptography.

Security for Cloud (Remote) Storage

- Confidentiality
  - Various encryption systems
- Integrity
  - Integrity verification protocols
- Availability
  - Redundancy
  - Error correcting code

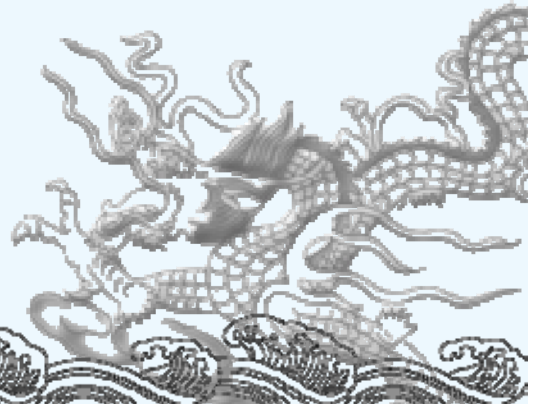


Integrity Verification

- Message digest
  - Naïve approach
  - No authenticated data integrity; Bandwidth wasting
  - Deterministic
- Provable data possession (PDP)
  - Authenticated data integrity
  - Probabilistic
- Proof of retrivability (PoR)
  - Authenticated data integrity & improved availability
  - Probabilistic

Lifecycle

- Repository (data) deployment
  - Generate tags
- Integrity verification
  - Challenge data integrity
  - Generate proof of storage
- (Optional) Repository evolution
  - Generate tags for modified part



Scenario for Integrity Verification

Third Party Verifier

2. Verifying Data Integrity

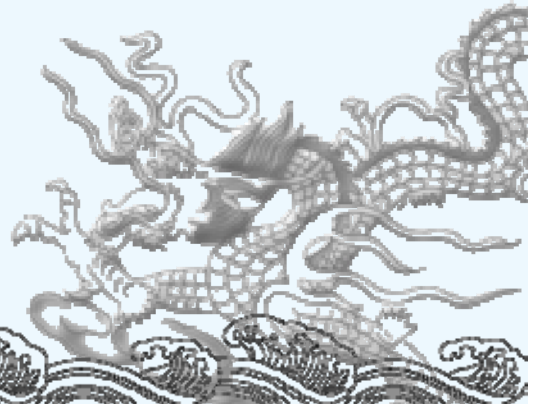
1. Deploying Data

Data Owner

Cloud Storage Provider

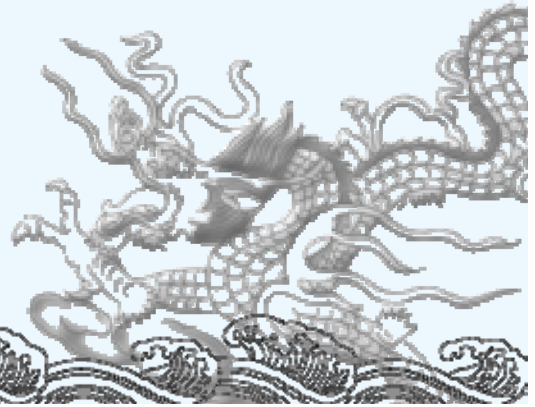
Issues

- Replay attack
  - The status of repository is not clear
- Performance
  - Slow verification
  - Even on personal computer
- Batch verification
  - Single user
  - Multiple users

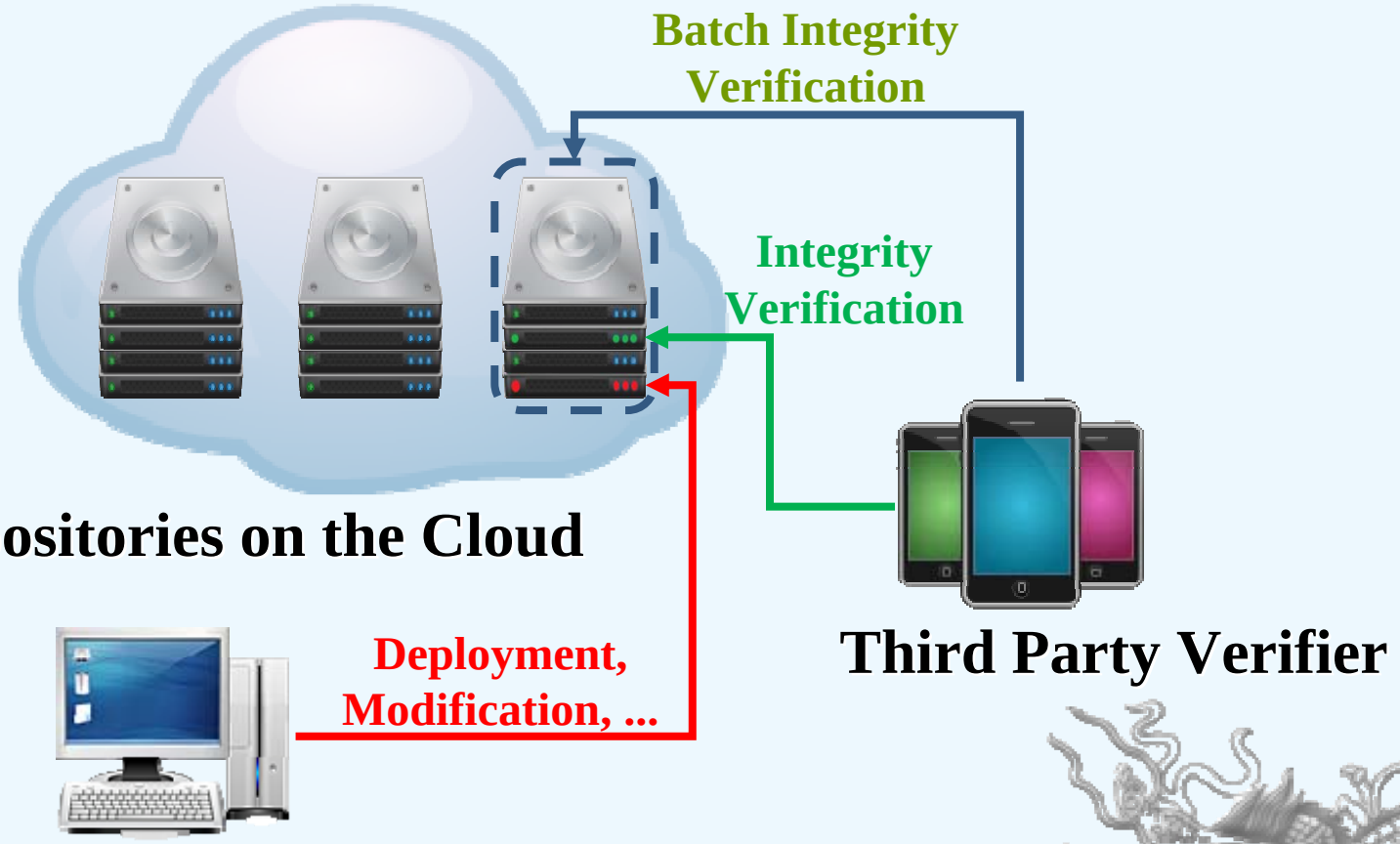


Approaches

- Replay attack
  - Revision number as timestamp
- Performance
  - Multiplication instead of exponential operations
- Batch verification
  - Repository as an single file



Scenario for Single User

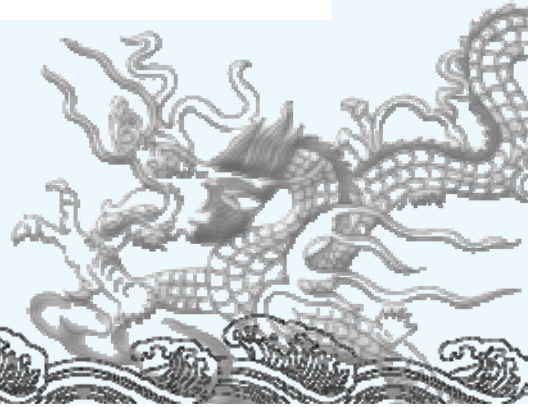


Bilinear Map

- Bilinear: $\forall h_1, h_2 \in G$, and $\forall a, b \in Z_p$, $e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$.
- Non-degenerate: $e(g, g) \neq 1$, for g is a generator of G.
- From above properties, we can get another: $\forall u, v_1, v_2 \in G$, $e(u, v_1 v_2) = e(u, v_1) \cdot e(u, v_2)$.

BLS Signatures

- A signer with the secret key: α, the public key g^{α}.
  - Signing: $\sigma_i = H(m_i)^{\alpha}$.
  - Verification: $e(\sigma_i, g) = e(H(m_i), g^{\alpha})$.



Tokens

- Wang *et al.* [1] :
  - Token Generation: $\sigma_i = \left(H(m_i^*) \cdot u^{m_i} \right)^{SK}$.
  - Verification: $e(\sigma, g) = e(u^{\mu} \prod H(m_i^*)^{v_i}, g^{SK}$
- Ours method:
  - Token Generation: $\sigma_i = \left(h^{H(m_i^*)} \cdot u^{m_i} \right)^{SK}$.
  - Verification: $e(\sigma, g) = e(u^{\mu} h^{\sum v_i H(m_i^*)}, g^{SK})$.

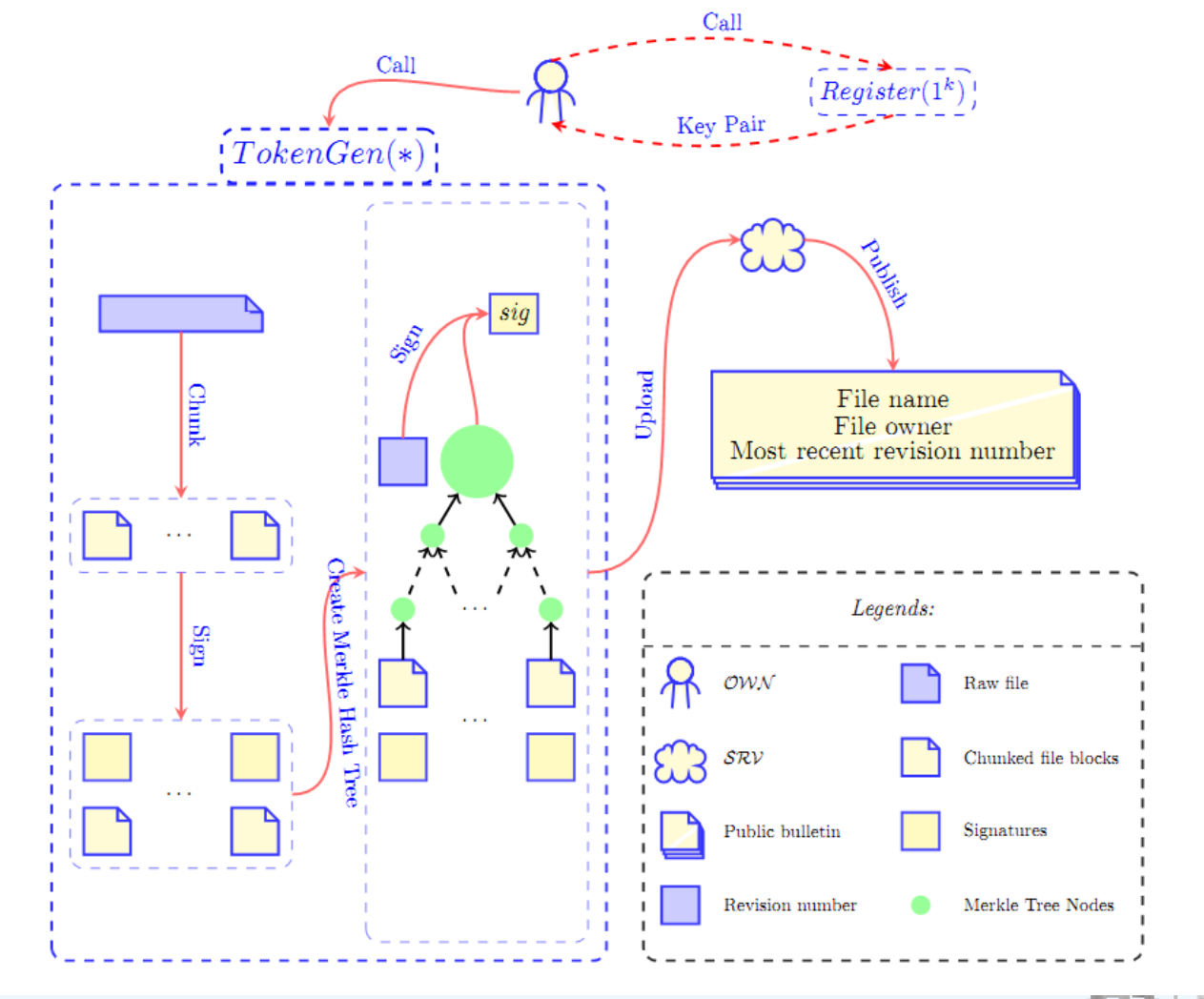
Security Concern

- **Exponential operations → multiplication**
  - Security flaw?
    - Ex: $(H(m))^{\alpha} \rightarrow (g^{h(m)})^{\alpha}$
    - $\left((g^{h(m)})^{\alpha}\right)^{h(m)^{-1}h(m')} = \left((g^{h(m')})^{\alpha}\right)$

Security Concern

- **Exponential operations → multiplication**
  - Security relies upon the CDH assumption
    - CDH assumption:
      - $Prob[g, g^{\alpha}, g^{\beta} \rightarrow g^{\alpha\beta}]$ is negligible
    - $\sigma \leftarrow \left((g^{\beta})^{h(m)} (g^{\gamma})^{x} \right)^{\alpha}$
    - $A_1(g, g^{\alpha}, g^{\beta}, g^{\gamma}, h(m), x) \rightarrow \sigma$
    - $A_2(\sigma, g^{\alpha}, \gamma, h(m), x) \rightarrow g^{\alpha\beta}$
      - $g^{\alpha\beta} = \left(\sigma \cdot \left((g^{\alpha})^{xy} \right)^{-1} \right)^{(h(m))^{-1}}$

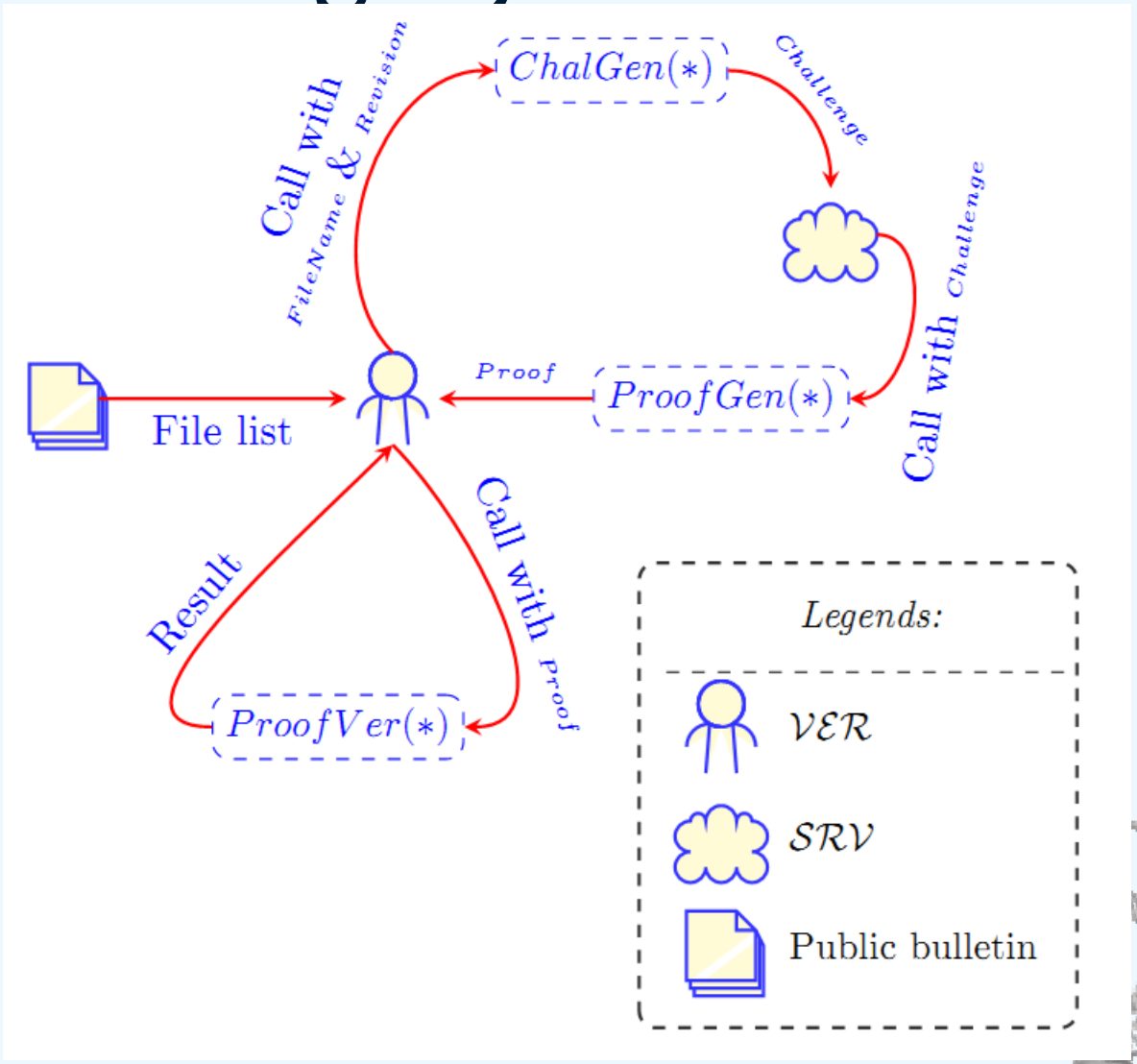
Repository Deployment



Repository Deployment

- **Parameter:**
  - $H_1: \{0,1\}^* \rightarrow G; H_2: \{0,1\}^* \rightarrow Z_p; H_3: \{0,1\}^* \rightarrow \{0,1\}^l$
  - $PK \leftarrow (V, \tilde{V}, h) = (g^{\alpha}, g^{\beta}, g^{\gamma}); SK \leftarrow (\alpha, \beta, \gamma)$
  - $u \leftarrow \tilde{V}^{H_2(ID||fn)}$
- **Tag generation**
  - For each $m_i \in F$
    - $\sigma_i \leftarrow \left(h^{H_2(H_2(m_i)||i)} \cdot u^{m_i}\right)^{\alpha}$
  - $\Phi \leftarrow \{\sigma_i\}_{1 \le i \le n}$
  - Generate MHT from $\{H_3(H_2(H_2(m_i)||i))\}_{1 \le i \le n}$
  - Sign on the root of the MHT:
    - $sig(R, ver) \leftarrow (H_1(R||ver))^{\alpha}$

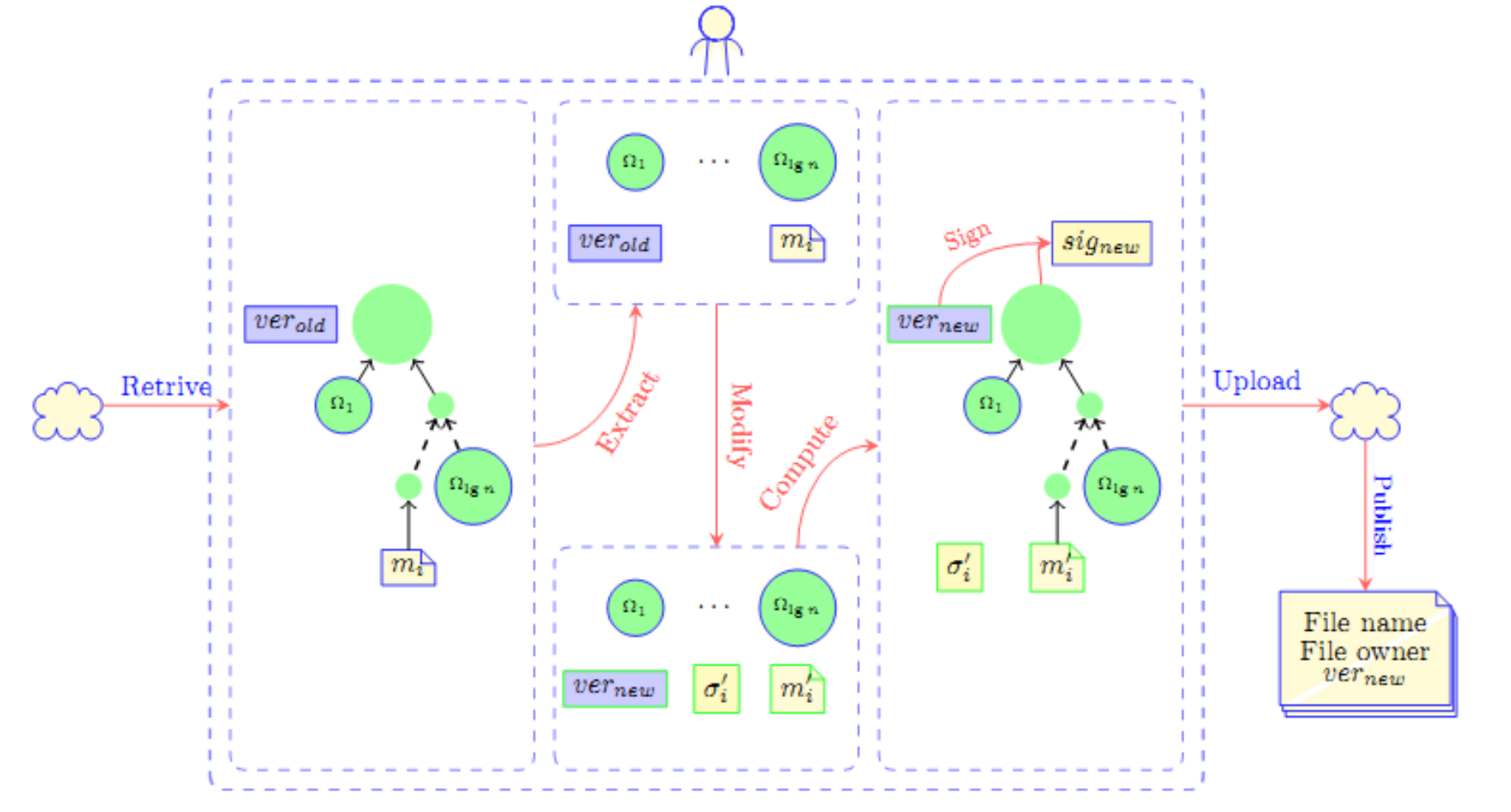
Integrity Verification



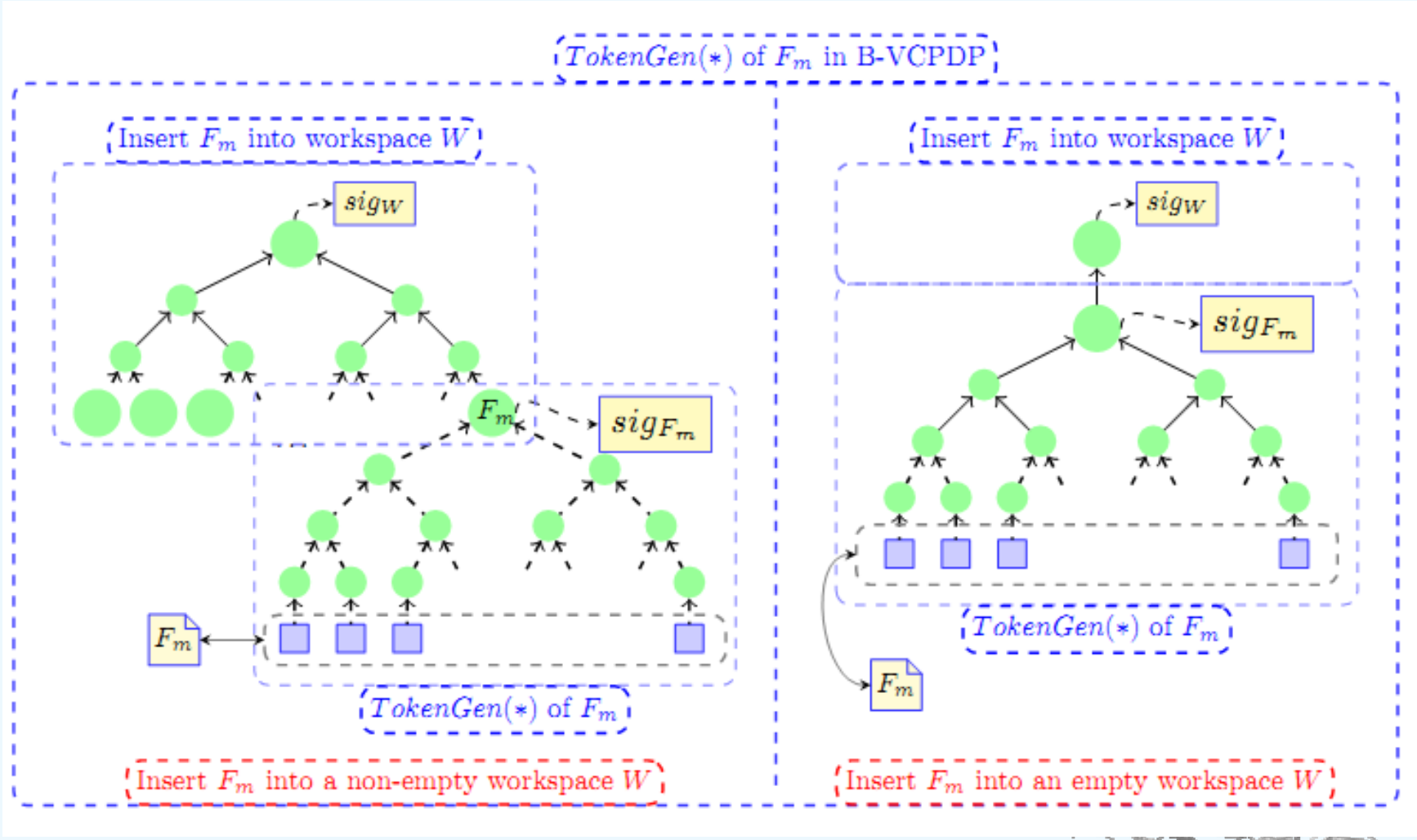
Integrity Verification

- **Proof:**
  - I is the set of challenged blocks
  - $\mu \leftarrow \sum_{i \in I} v_i m_i \, ; \sigma \leftarrow \prod_{i \in I} \sigma_i$
  - Ω_I is the set of nodes to rebuild the root
- **Verify:**
  - Rebuild root R'
  - Check if $e(sig(R, ver), g) = e(H_1(R' || ver), V)$
  - $\rho \leftarrow \sum_{i \in I} H_2(H_2(m_i) || i) \cdot v_i$
  - Check if $e(\sigma, g) = e(h^{\rho} u^{\mu}, V)$

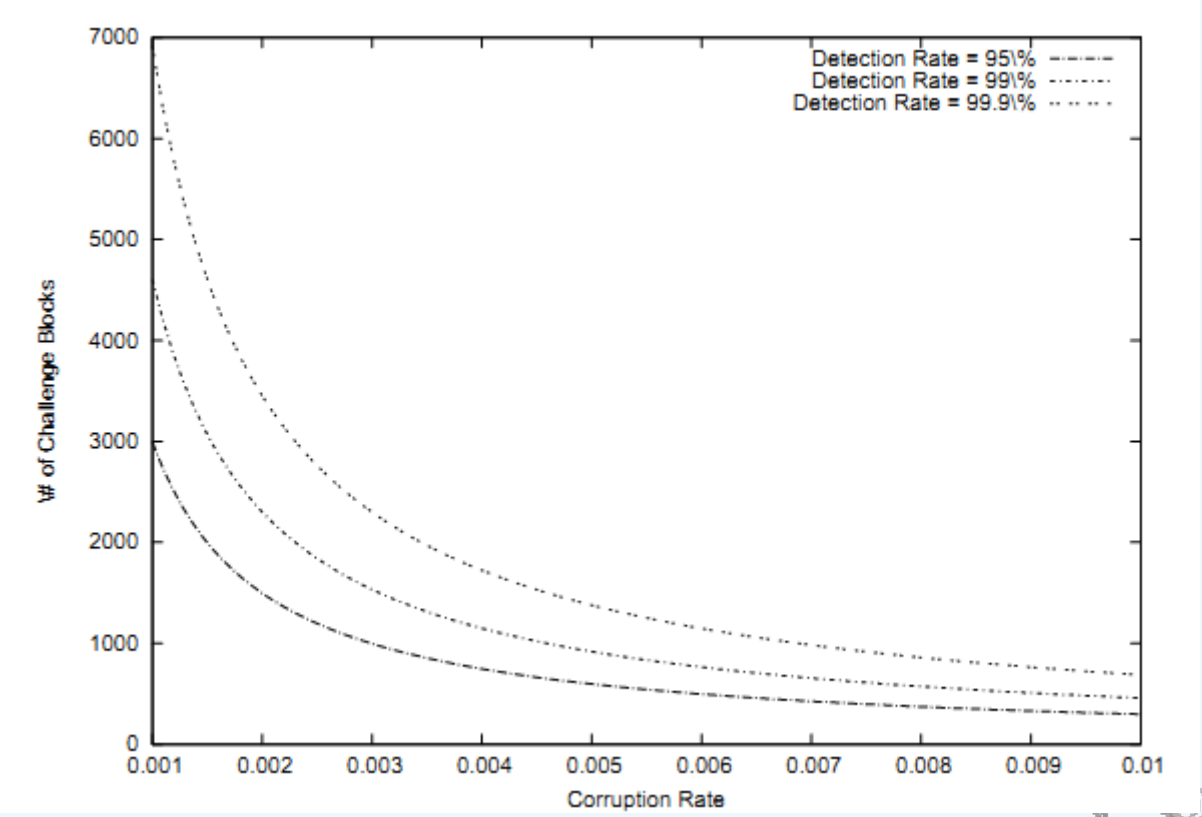
Repository Evolution



Batch Verification for Single User



Probabilistic Detection



Number of blocks needed to fulfill certain detection rate under various data corruption rate

Probabilistic Detection

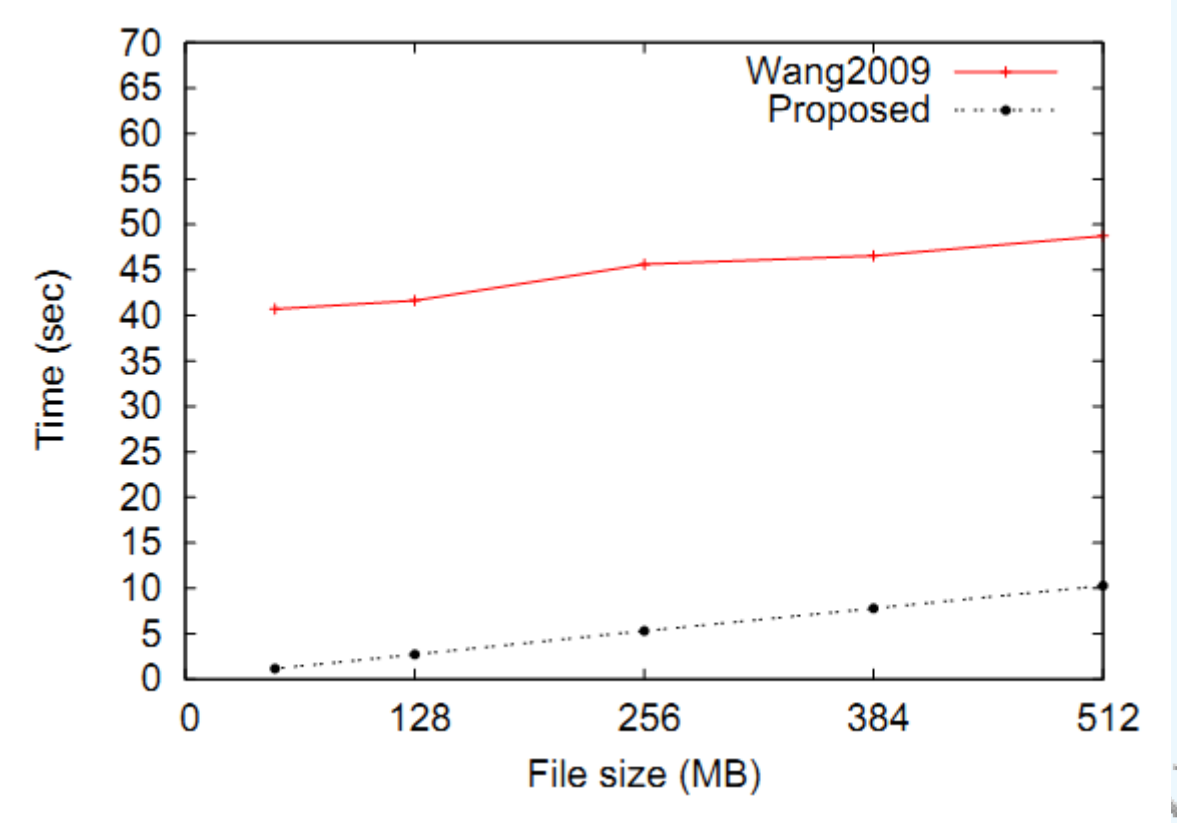
- Check points

# of Challenged Blocks	Detection Rate	Data Corruption Rate
300	95%	1%
460	99%	1%
4610	99%	0.1%
6910	99.9%	0.1%

Benchmarks

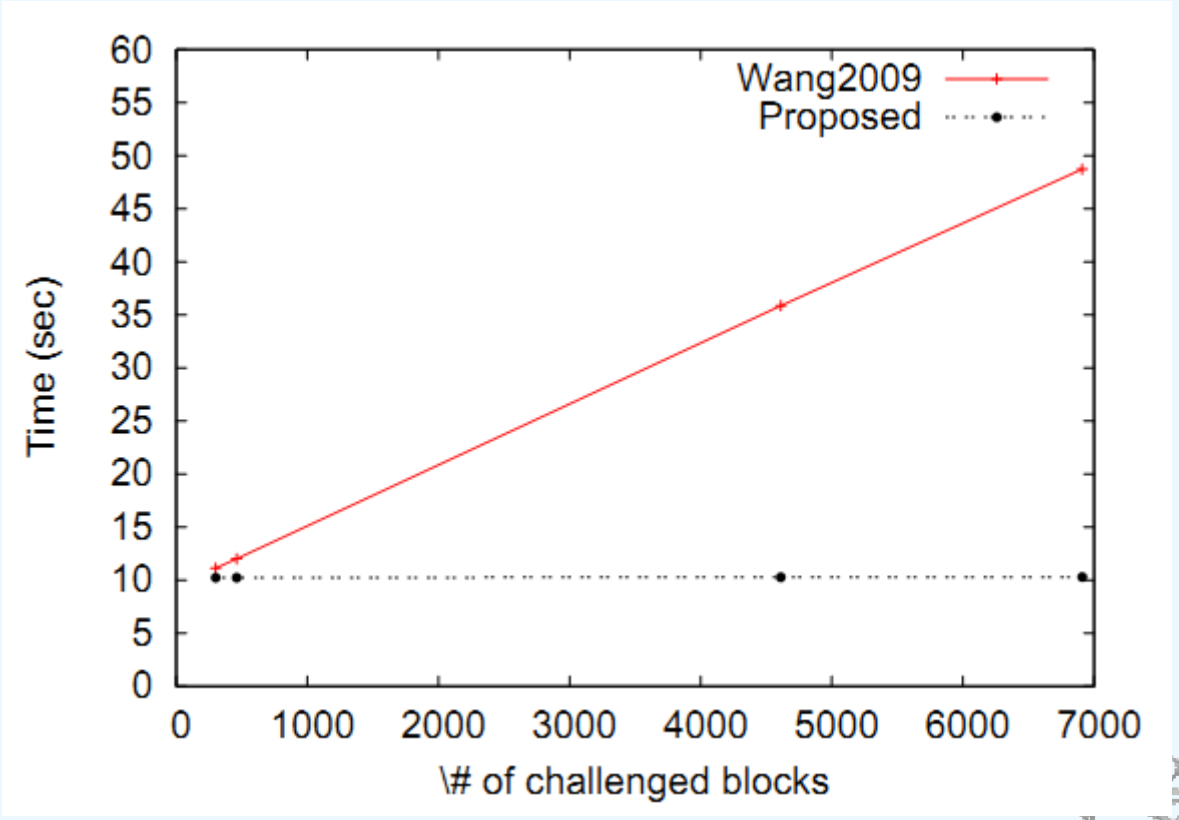
Operation	Personal Computer	Smart Phone
Multiplication Z_p (ab)	$5 \times 10^{-6} sec$	$6 \times 10^{-5} sec$
Multiplication G (gh)	$1.45 \times 10^{-4} sec$	$1.26 \times 10^{-3} sec$
Exponential G (g^a)	$3.29 \times 10^{-2} sec$	$2.64 \times 10^{-1} sec$
Pairing $G \times G \rightarrow G_T$ $(e(g^a, h^b))$	$3.84 \times 10^{-2} sec$	$5.41 \times 10^{-1} sec$

Verification Time



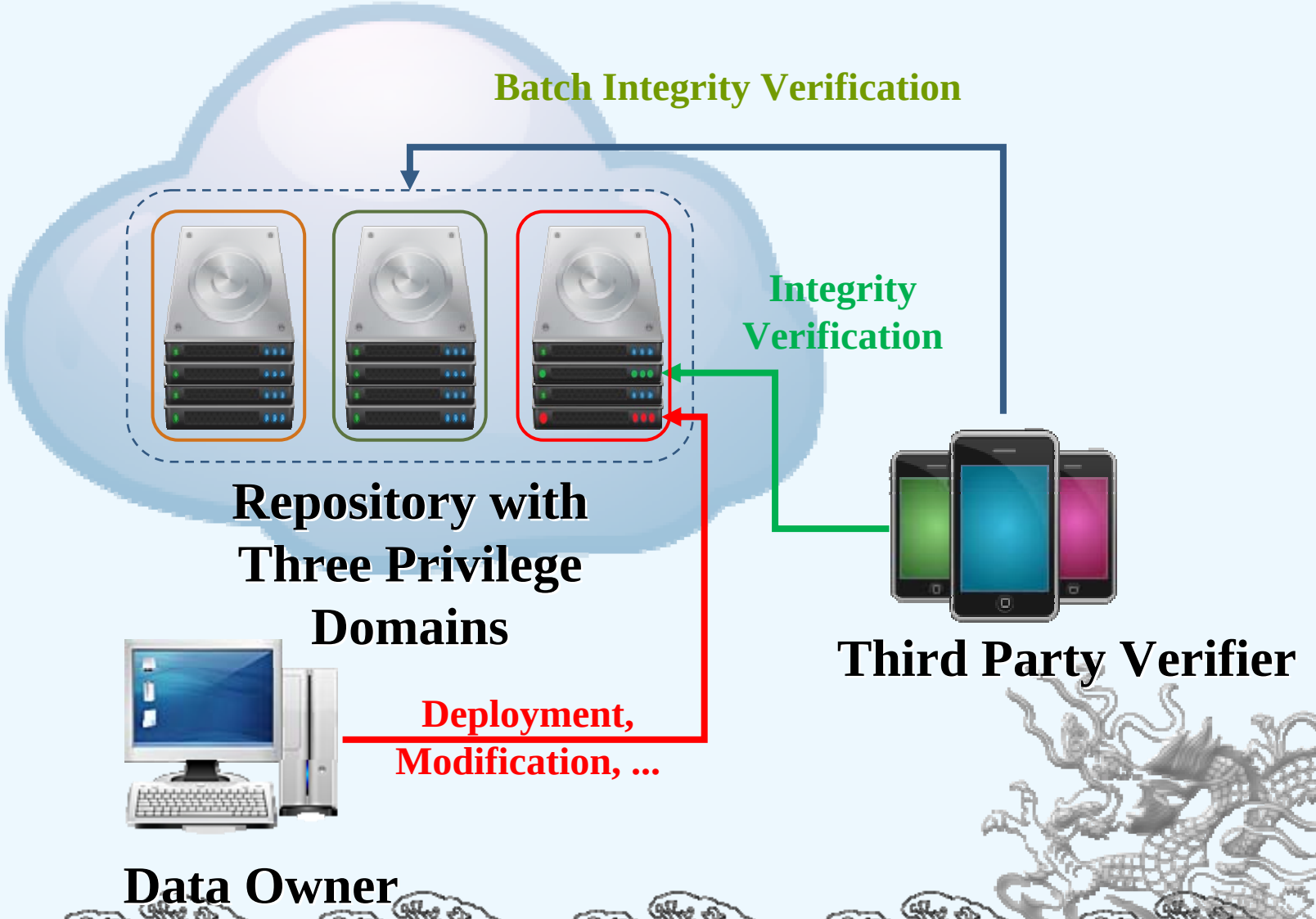
Client-side verification time with 6910-block challenge

Verification Time



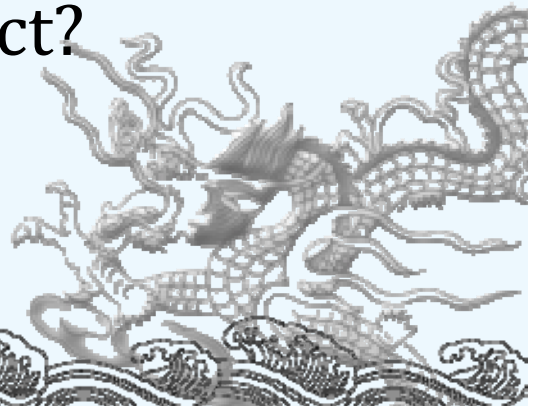
Client-side verification time with 512-megabyte file

Scenario for Multiple Users



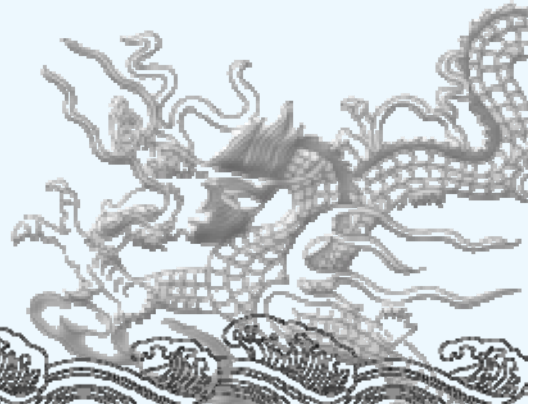
From Single User to Multiple Users

- Access control
  - Who can commit modifications of a certain part?
- Batch verification
  - Verify integrity across different users' data
- Race condition
  - Concurrent write of the same project?

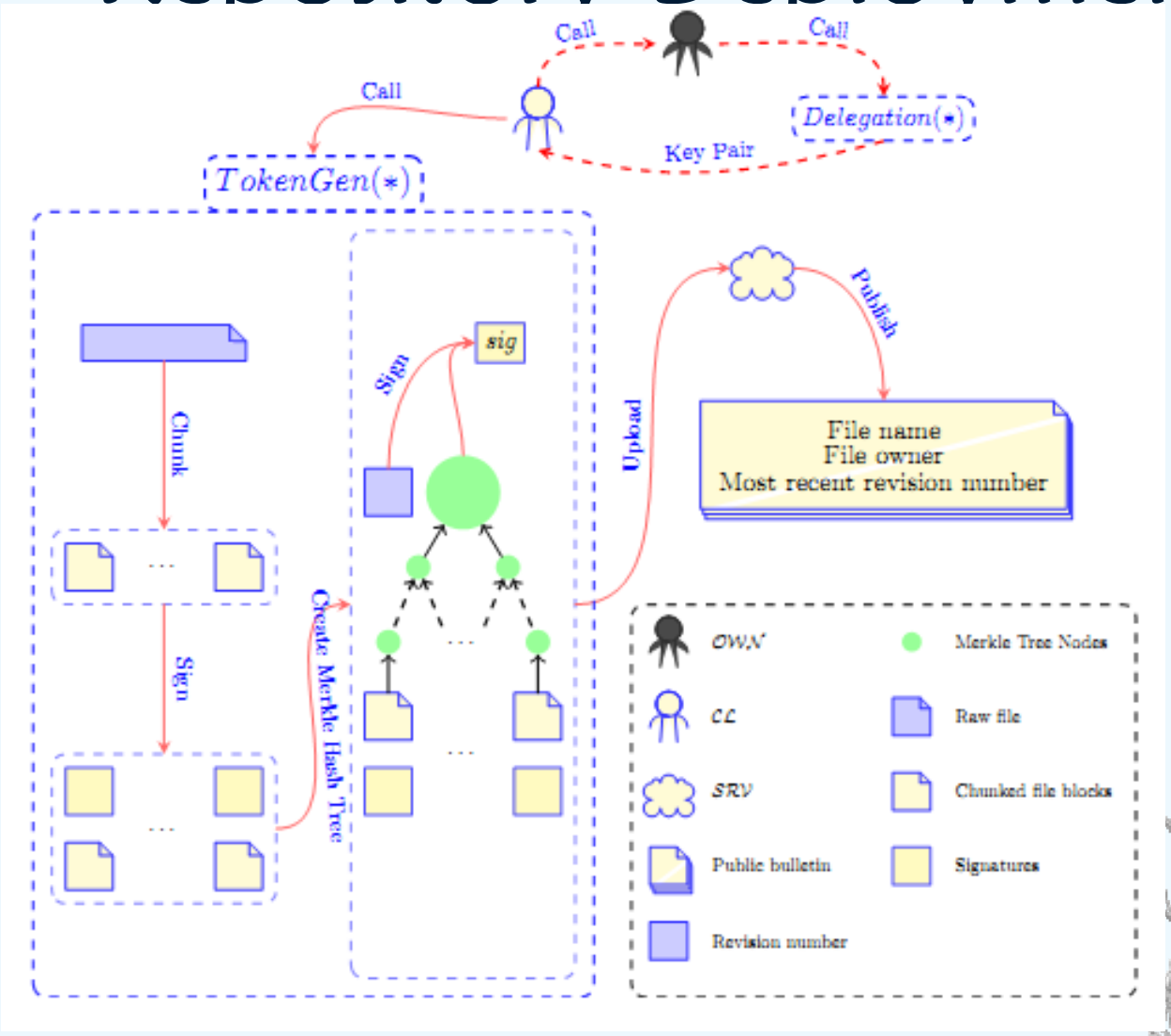


Approaches

- Access control
  - Multiple authority
    - Hierarchical
- Batch verification
  - Repository as a single file
- Race condition
  - Branching-and-merging



Repository Deployment



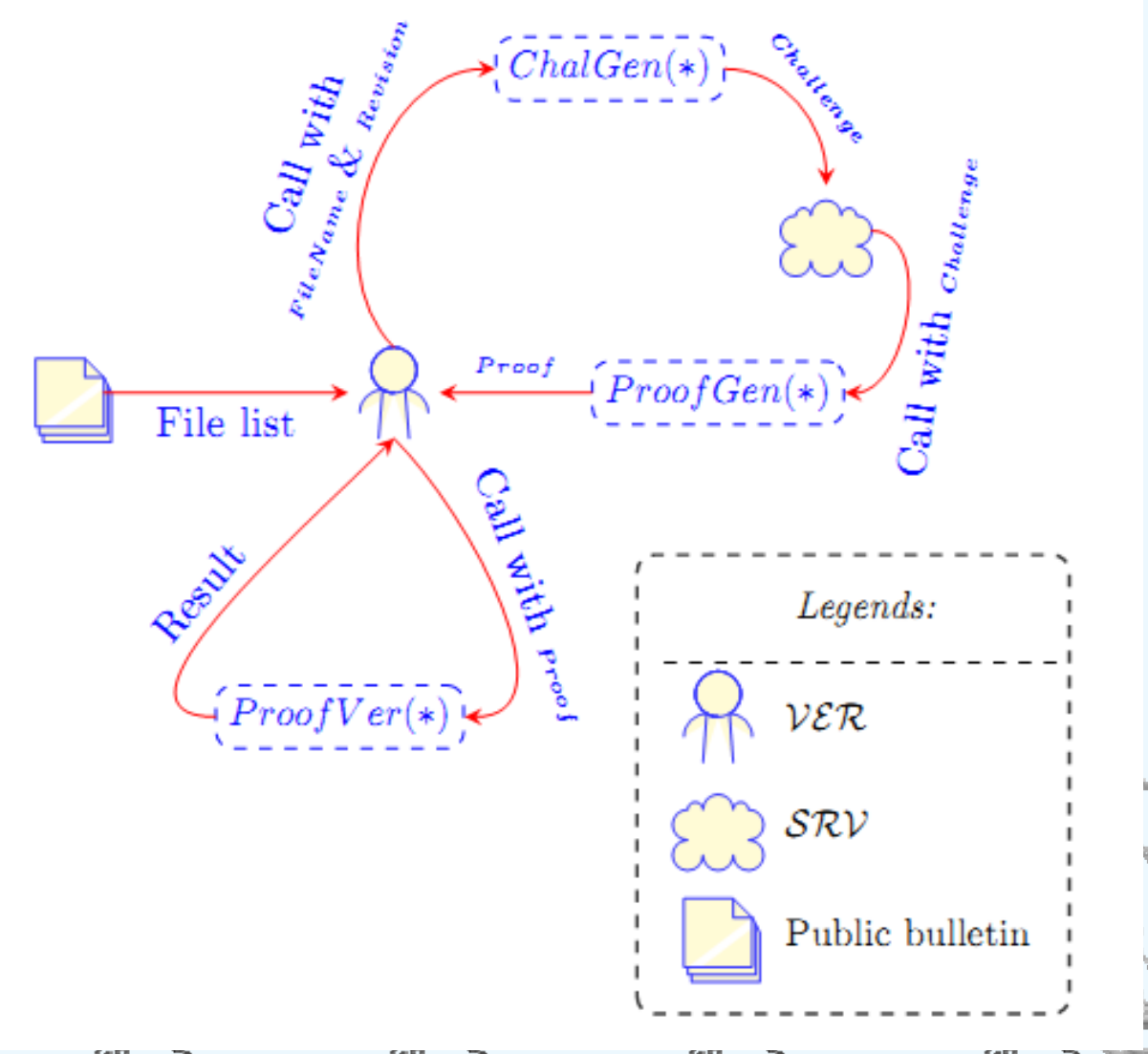
Key Delegation

1) $\eta_{\mathcal{CL}}, \theta_{\mathcal{CL}} \xleftarrow{R} Z_p^*$
2) $\pi_{\mathcal{CL},i} \leftarrow \alpha\pi_i + \eta_{\mathcal{CL}}, \forall i \in \mathcal{P}_{\mathcal{CL}}$, where $\mathcal{P}_{\mathcal{CL}}$ is the set of privilege domains which are granted to $\mathcal{CL}$
3) $\tilde{W}_{\mathcal{CL},i} \leftarrow g^{-\eta_{\mathcal{CL}}\pi_i^{-1}}, \forall i \in \mathcal{P}_{\mathcal{CL}}$
4) $SK_{\mathcal{CL}} \leftarrow \{\theta_{\mathcal{CL}}, \pi_{\mathcal{CL},i}\}_{i\in\mathcal{P}_{\mathcal{CL}}}$
5) $PK_{\mathcal{CL}} \leftarrow (W_{\mathcal{CL}}, \tilde{W}_{\mathcal{CL}}, \overline{W}_{\mathcal{CL}}) = (g^{\theta_{\mathcal{CL}}}, \{\tilde{W}_{\mathcal{CL},i}\}_{i\in\mathcal{P}_{\mathcal{CL}}}, H_1(W_{\mathcal{CL}}||\tilde{W}_{\mathcal{CL}})^{\alpha})$

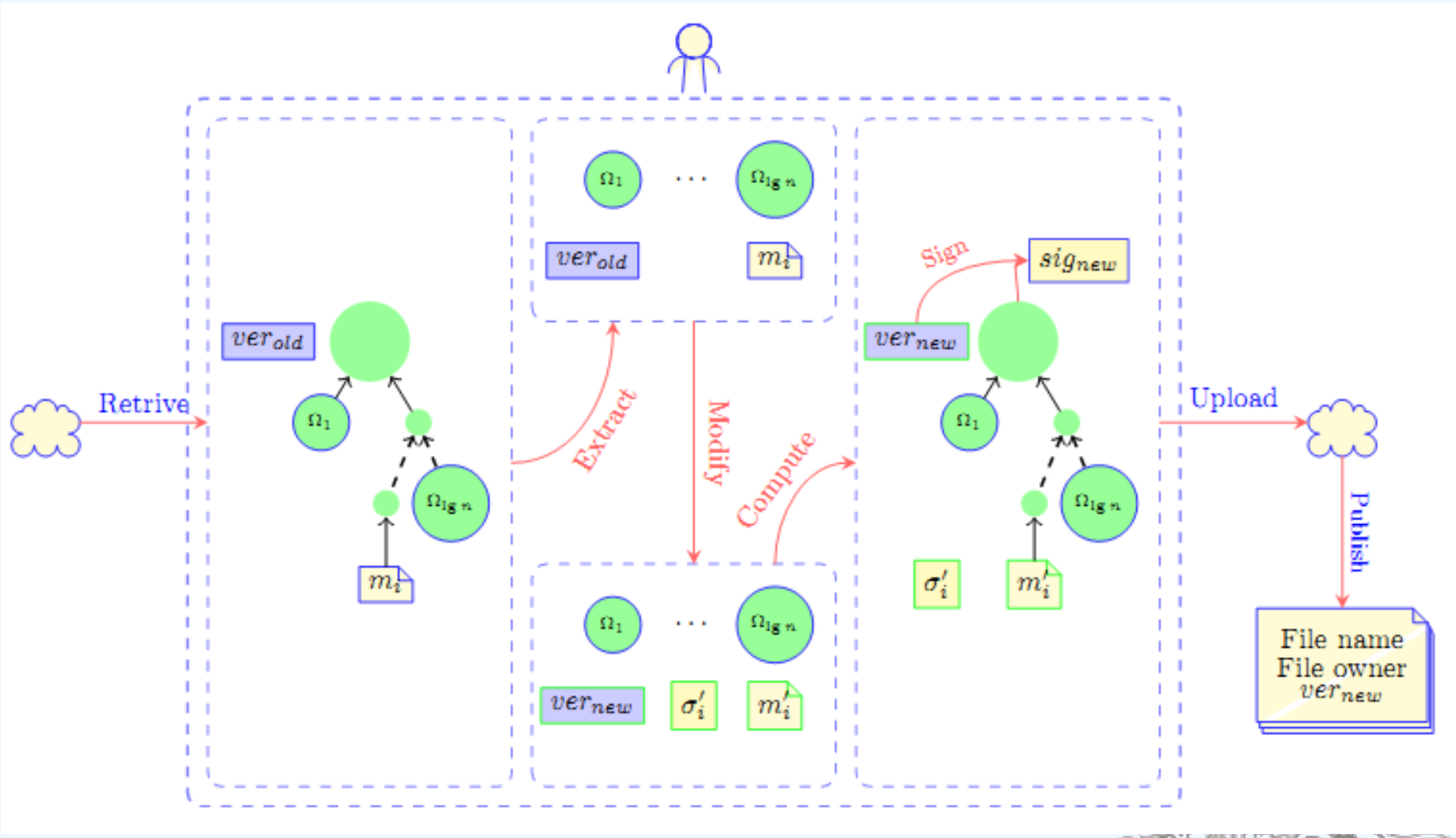
Tag Generation

1) $\sigma_i \leftarrow \left(h^{H_2(H_2(m_i)||i)} \cdot u^{m_i}\right)^{\pi_{\mathcal{CL},x}}$
2) $\Phi \leftarrow \{\sigma_i\}_{1 \le i \le n}$
3) Generate an MHT, $\mathcal{T}_B$, from $\{H_3(H_2(H_2(m_i)||i))\}_{1 \le i \le n}$
4) $sig_{\mathcal{CL}}(R, ver) \leftarrow (H_1(R||ver))^{\theta_{\mathcal{CL}}}$, where R is the root of $\mathcal{T}_B$
5) $\mathcal{CL} \rightarrow \mathcal{SRV}$: $\{fn, u, ver, F, \Phi, sig_{\mathcal{CL}}(R, ver)\}$, where fn is the file name

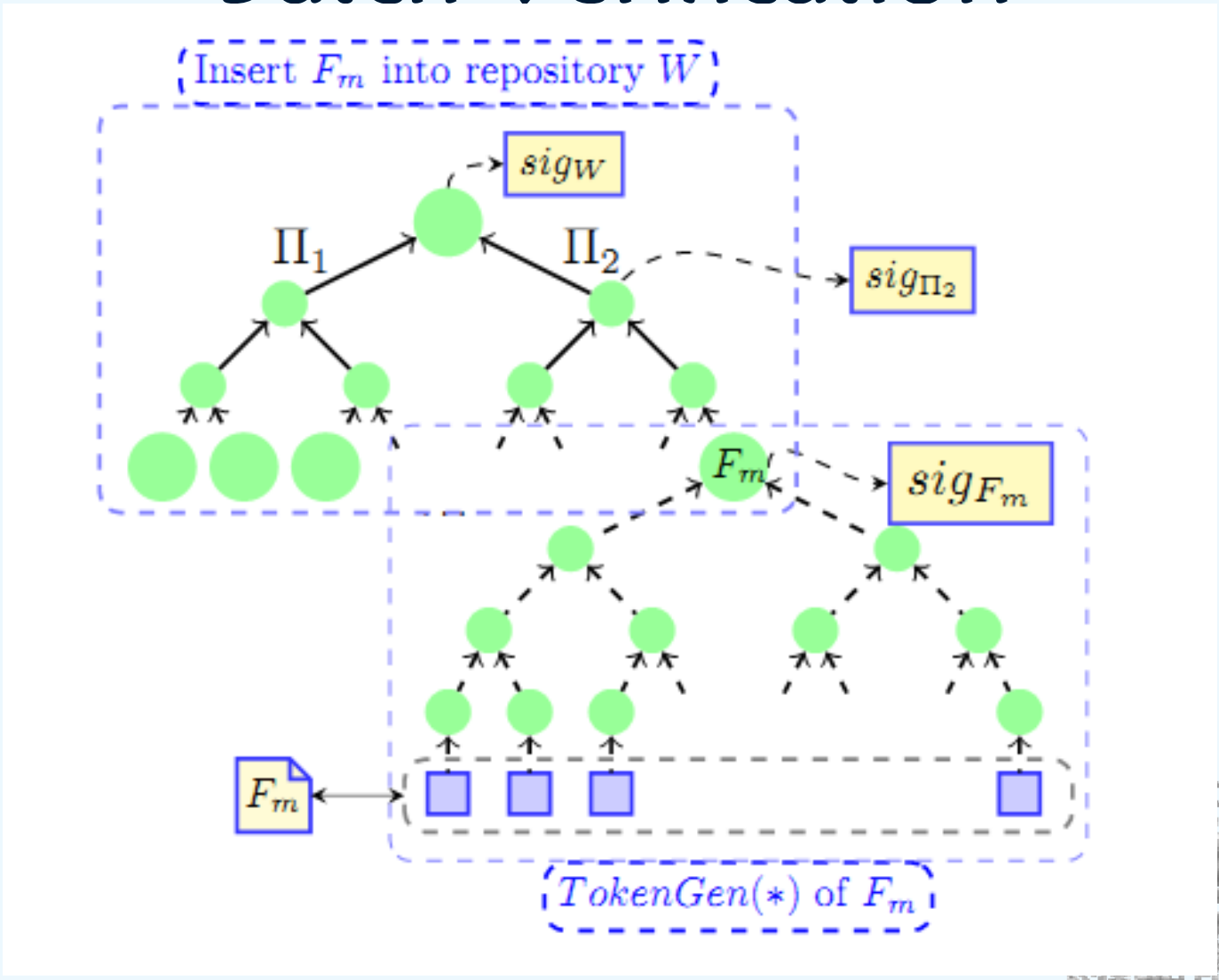
Integrity Verification



Repository Evolution

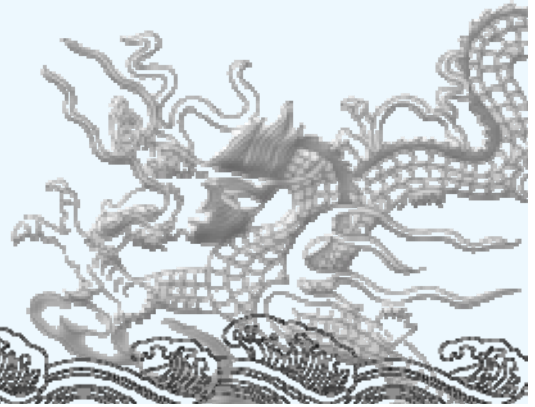


Batch Verification

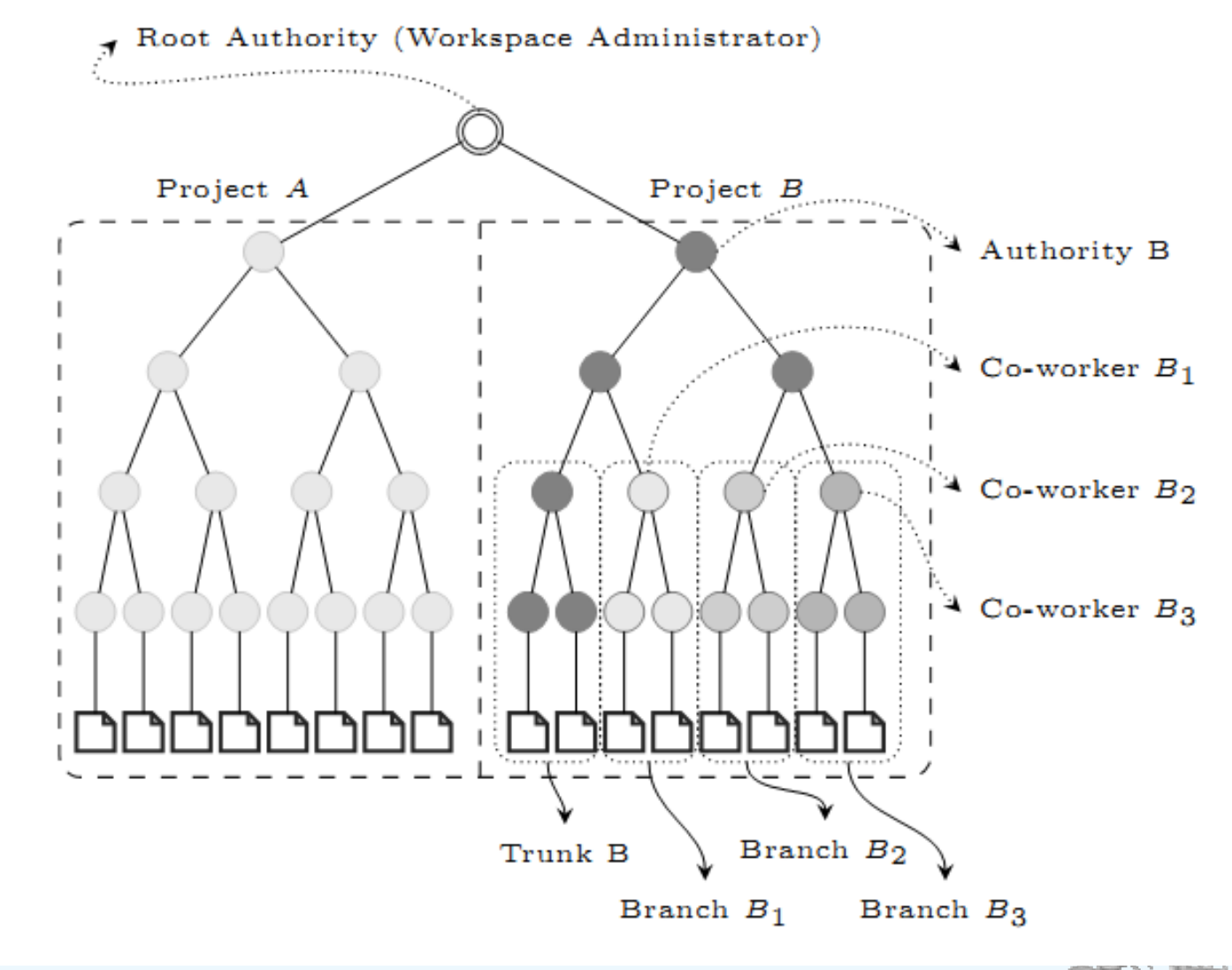


Branching-and-Merging

- Before modify shared data
  - Copy to one's own privilege domain (branching)
- After finish the modification
  - Coordinate with other collaborators
  - Write the modifications to the trunk (merging)

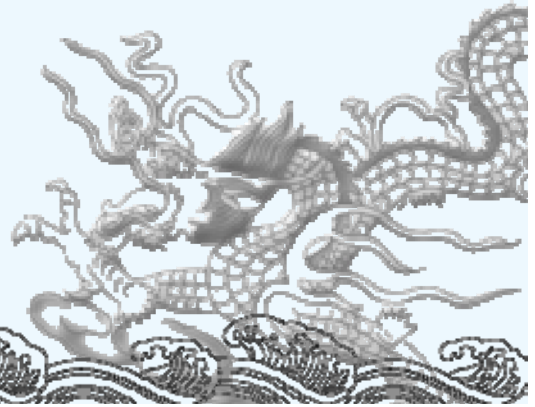


Branching-and-Merging (Example)



Conclusion

- Efficient integrity verification
  - Can even run on smart phone!
- Batch verification
  - Convenient for verifiers
  - Suitable for online co-working

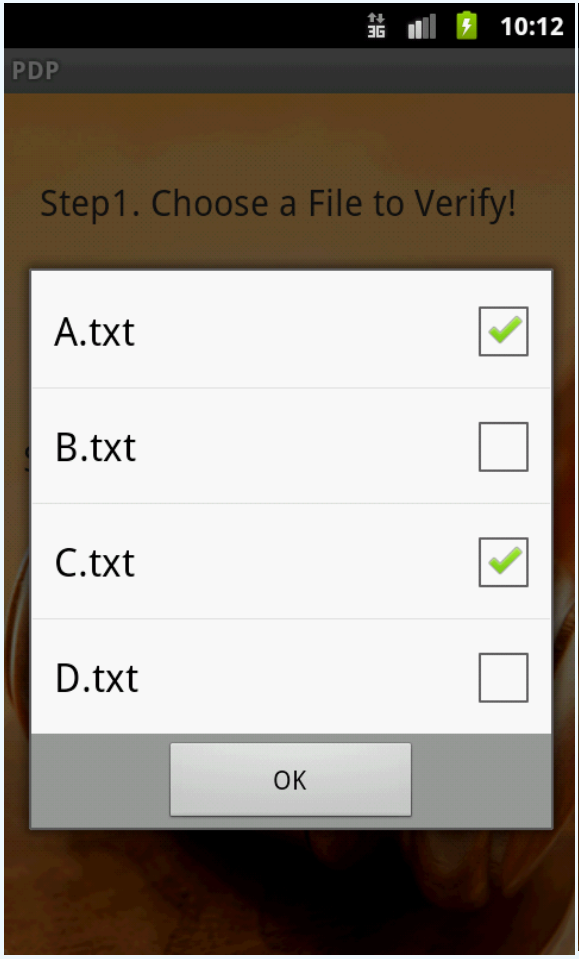


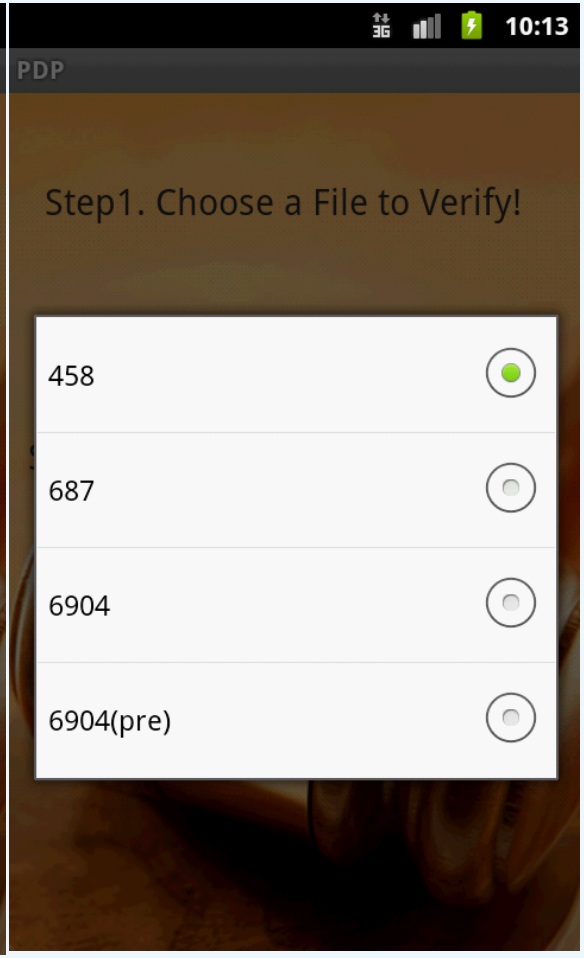
科学技術振興機構
Japan Science and Technology Agency

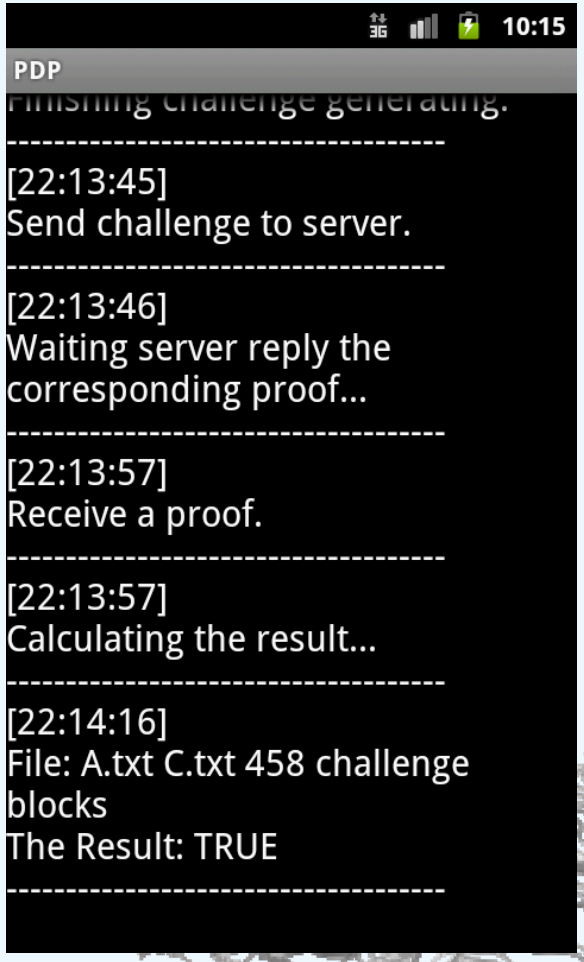
行政院國家科學委員會
National Science Council

Thank You

Appendix


10:12
PDP
Step1. Choose a File to Verify!
A.txt
B.txt
C.txt
D.txt
OK


10:13
PDP
Step1. Choose a File to Verify!
458
687
6904
6904(pre)


10:15
PDP
[22:13:45]
Send challenge to server.
[22:13:46]
Waiting server reply the corresponding proof...
[22:13:57]
Receive a proof.
[22:13:57]
Calculating the result...
[22:14:16]
File: A.txt C.txt 458 challenge blocks
The Result: TRUE