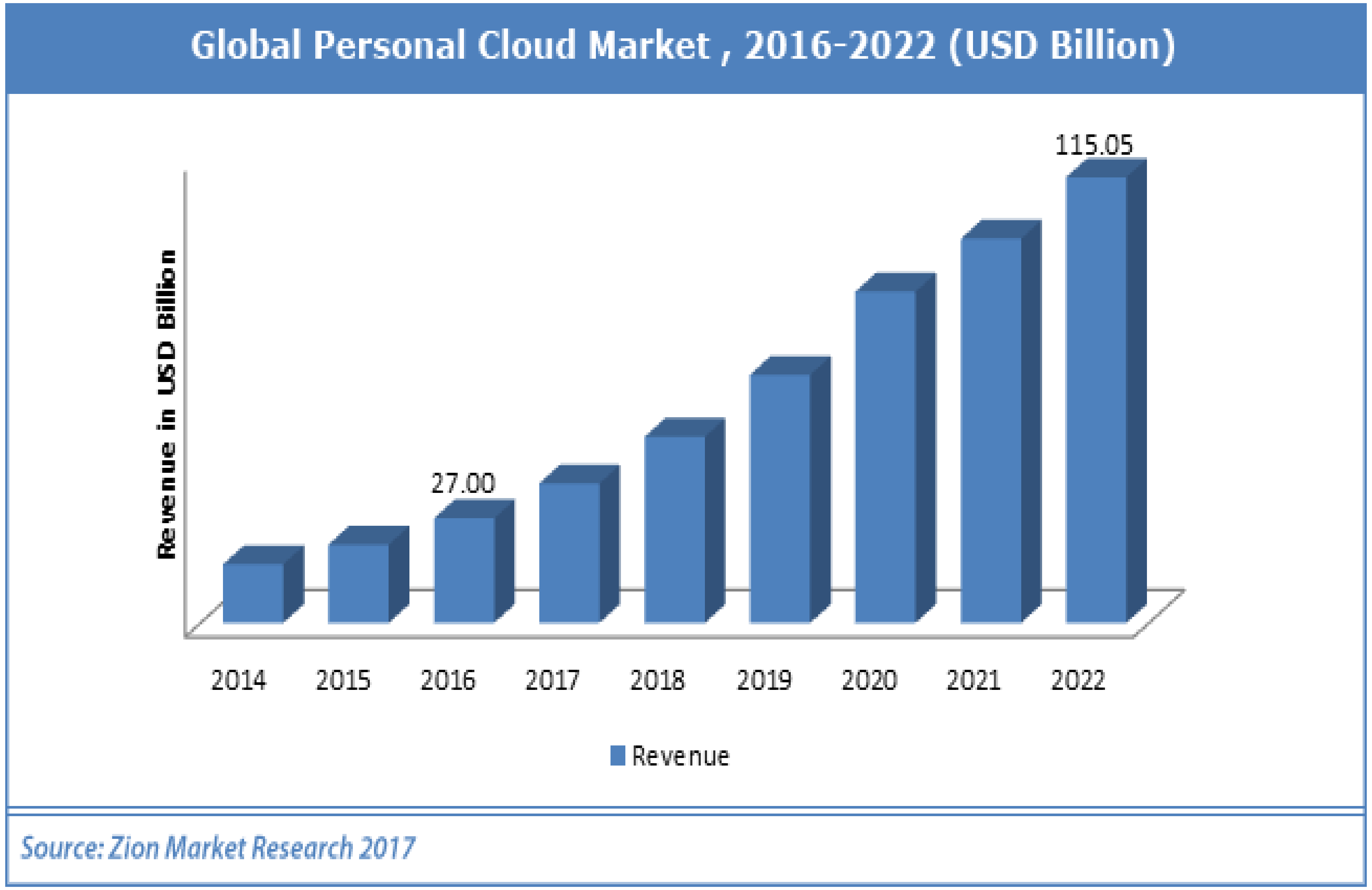
Towards Privacy-assured and Lightweight On-chain Auditing of Decentralized Storage

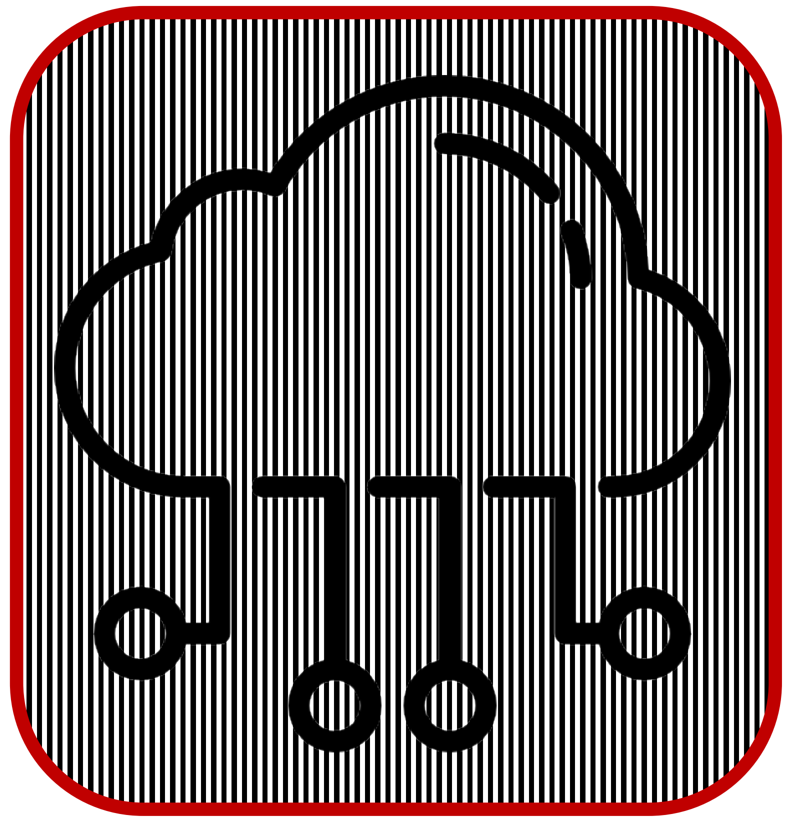
[1]Yuefeng Du, [1]Huayi Duan, [1]Anxin Zhou, [1]Cong Wang, [2]Man Ho Au, [3]Qian Wang

[1]City University of Hong Kong
[2]University of Hong Kong
[3]Wuhan University

Ubiquitous cloud storage



However…



Centralized

What’s been done inside?

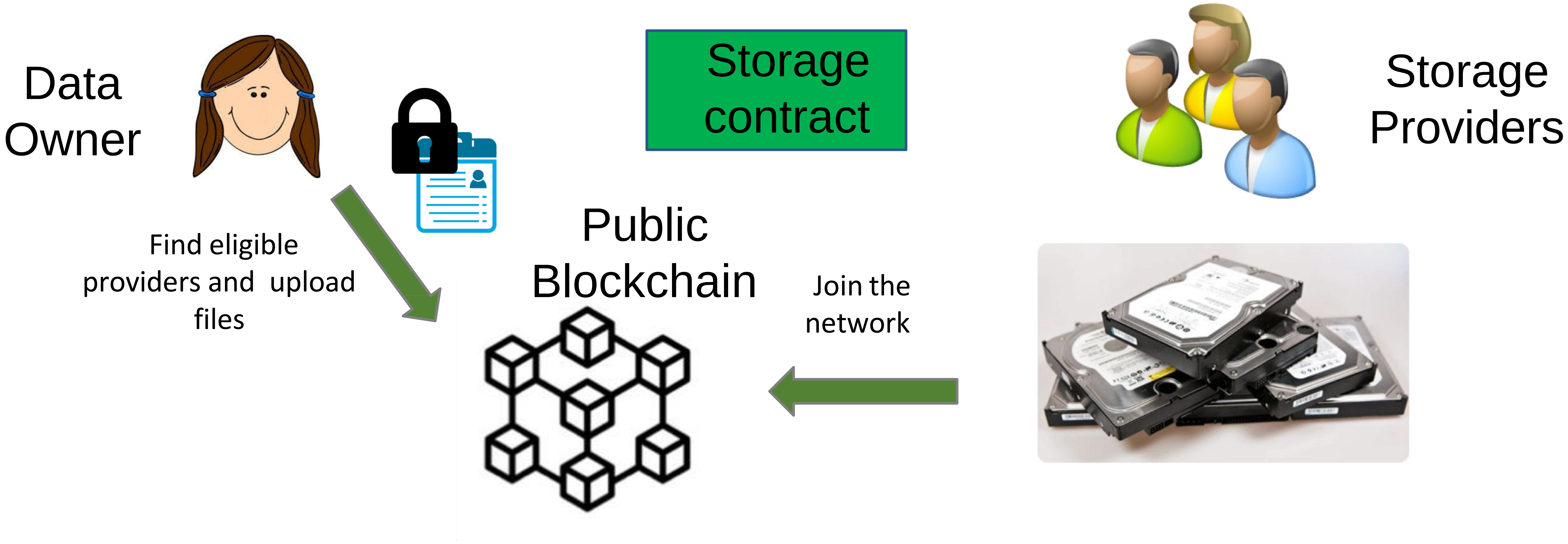
- Data privacy concerns
- Opaque service model
- Blind trust based SLA, e.g., data integrity and data availability
- …

Active Research on extending visibility inside cloud

- Proof of Storage
- Proof of Data Encryption
- Proof of Data Redundancy
- Proof of Ownership
- Cryptographic Database System
- Confidential Computing
- …

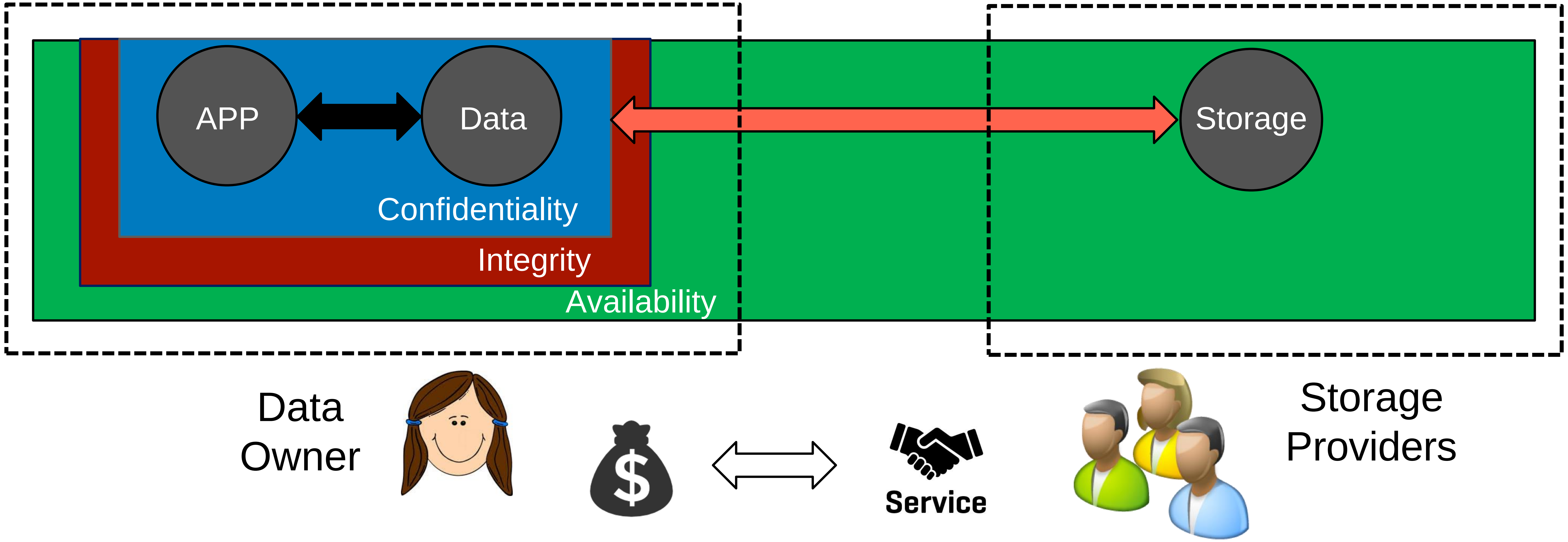
Yet, little incentive to adopt all

Growing interest in decentralized storage



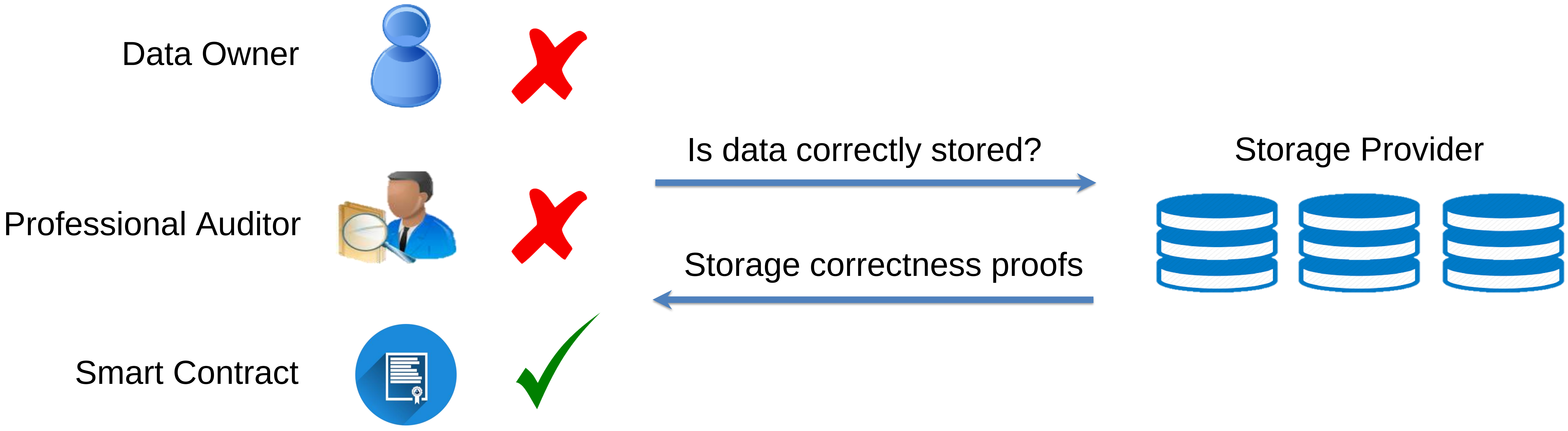
- Sharing economy paradigm
  - Individual providers rent out unused storage for rewards
  - Bodes well a billon-dollar marketplace

General picture of data outsourcing procedures



- An alternative to cloud storage
  - Built-in encrypted storage & data integrity guarantee
  - Transparent redundancy/replication for availability
    - ➢ Need continuous auditing to ensure storage services?

Storage auditing



- A challenge-response protocol for storage integrity/retrievability assurance

Primitives for storage auditing

Proofs of Retrievability (PoR) [Juels-Kaliski ‘07]

- An *efficient* audit protocol between client & server.
- A server that passes the audit must *know* all of the client data.

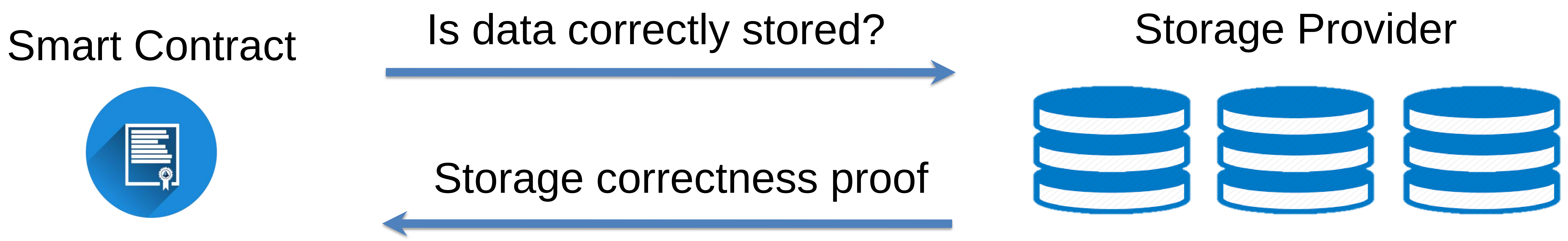
Efficiency: client and server computation is polylog in size of data.

Knowledge: formalized using an *extractor* (proof-of-knowledge [GMR85]).

Related notions:

- Sub-linear authenticators [Naor-Rothblum ‘05]
- Proofs of data possession [Ateniese et al. ‘07], e.g., Merkle tree construction

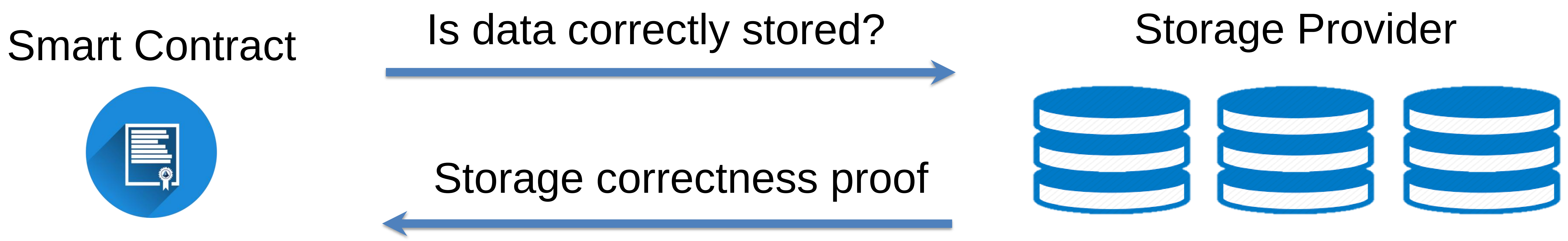
Continuous auditing for decentralized storage



- Starting from PoR/PDP, latest efforts as Proof of Storage-time [NDSS2020]
  - Formalizing continuous auditing, a generic extension of PoR/PDP
  - The instantiation is yet to be satisfactory nor practical:
    - Stateful with bounded usage
    - Large prover cost*
    - Intrinsically not friendly to dynamics

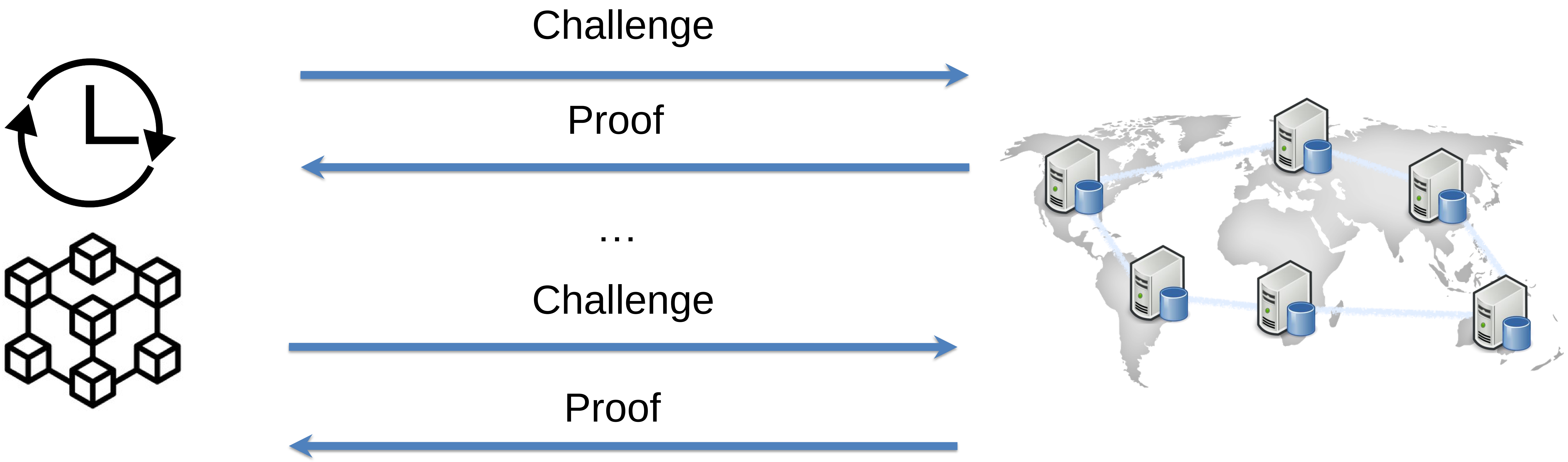
*Intentional design choice for a security consideration

Continuous auditing for decentralized storage



- We focus on a concrete auditing design in the context of DSN
  - Preventing threats that exploit on-chain proofs
  - Concrete efficiency in practical settings
  - Possible adoption to complement prior arts in continuous auditing
  - More friendly to potential dynamics support

Periodical and transparent auditing



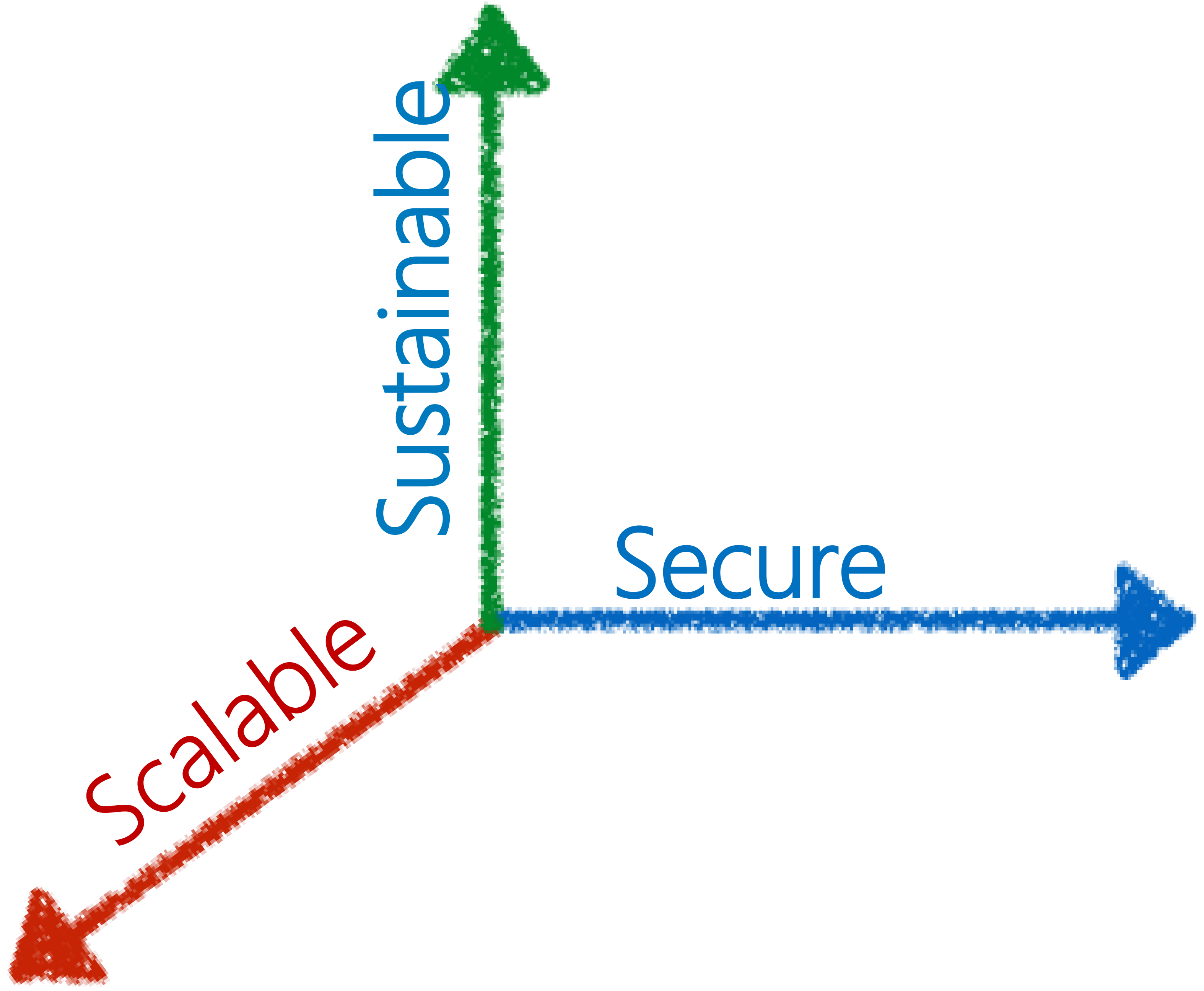
- Audit history stored on the blockchain
- Natural fit in the incentive system
- Technically strengthen SLA assurance

However, …

Immed. Chal. #1 When transparency meets extractability …

- Audit history on the blockchain may be abused to recover partial data
  - Any off-chain adversaries can abuse on-chain data stealthily
  - Proofs on chain must not reveal bits for data recovery, regardless of data encryption (Finck, Michèle. "Blockchains and data protection in the European union." *Eur. Data Prot. L. Rev.* 4 (2018): 17.)

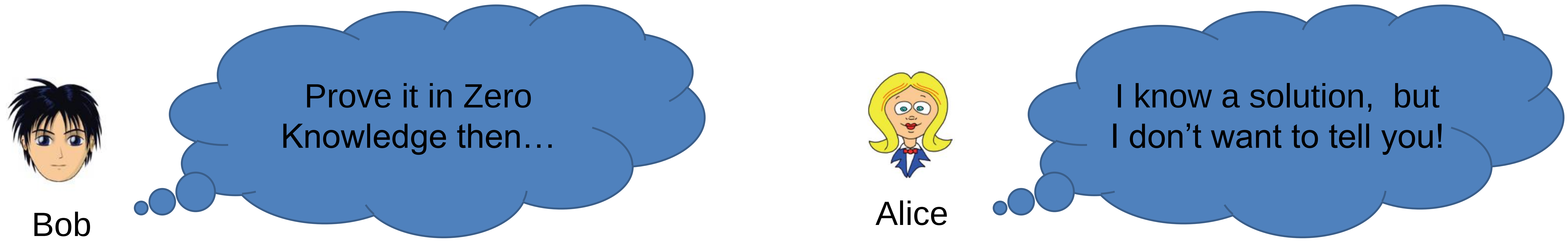
Immed. Chal. #2 Concrete efficiency is critical



- As on-chain proof verification is done by each miner, thus we need
  - Succinct proof
  - Quick verification
- Ideally, reduce overall cost as far as possible
  - Data preprocessing
  - Prover cost
  - More...

Begin with zero knowledge auditing

- Revealing nothing but the correctness of auditing proofs
  - Adopt generic frameworks over any storage auditing design
  - Apply customized approach on specific storage auditing scheme



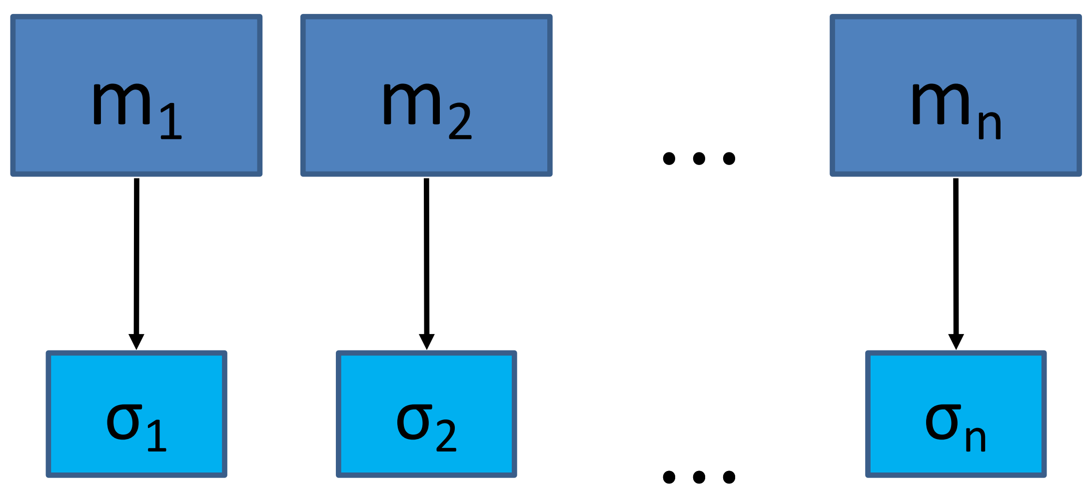
Generic approach not yet practical

- ZK-SNARK (generic ZKP framework) wrap-up over Merkle tree for zero knowledge auditing
  - In a Merkle tree, with root R, we can verify any leaf nodes
  - Verification: h(h(a, h(x), c) = R, where h is a cryptographic hash function
- Large overhead yet to be overcome, and hardly scalable

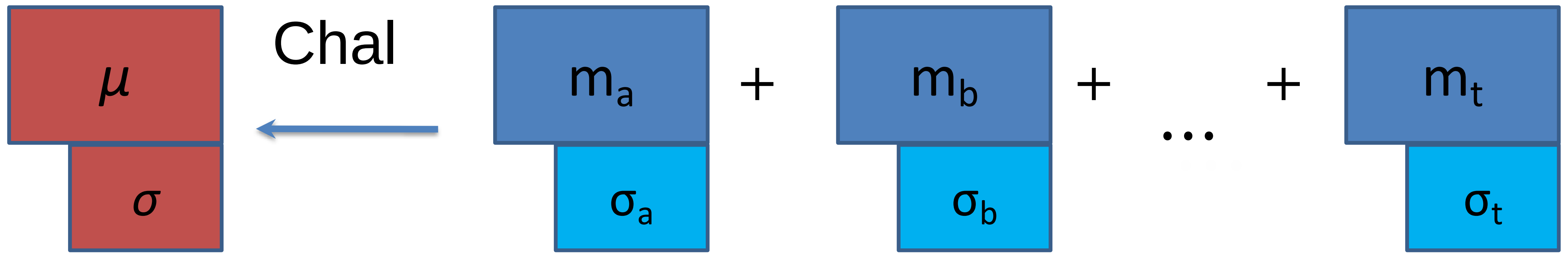
	File Info.	Pre-process†			Proof Generation			Verification
	Size	Time	Param. size	# Constraints	Time	Memory	Size	Time
Strawman solution*	1 KB	260 s	150 MB	3×10^5	30 s	$\sim$ 300 MB	**384 bytes**	**30 ms**

We resort to customized approach

- Homomorphic Linear Authenticator (HLA)
  - Generate authenticator (signature) for each data block for verification.



  - Data blocks and authenticators can be aggregated



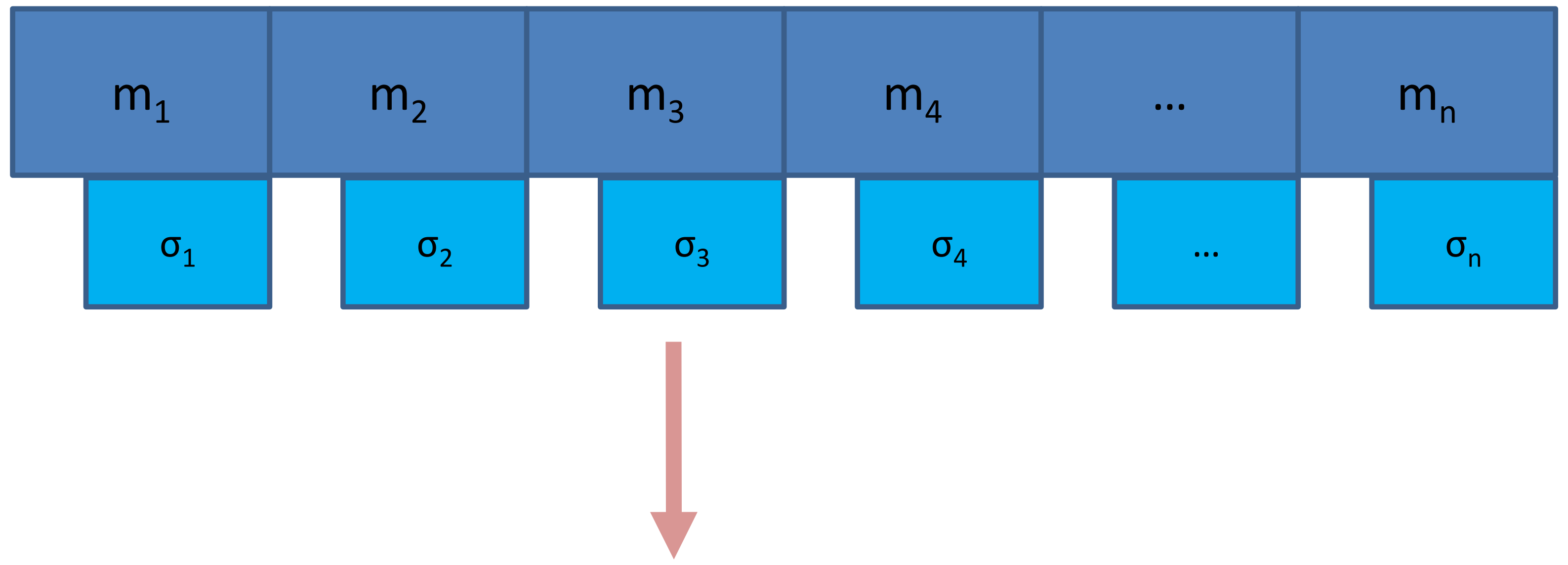
We resort to customized approach

A quick exemplary illustration

- Random masking* for ZK storage auditing

Friendly to algebraic operations

- Small computational overhead
- Small increase in proof size



$$\mu = v_1 m_1 + v_5 m_5 + v_6 m_6 + v_8 m_8$$

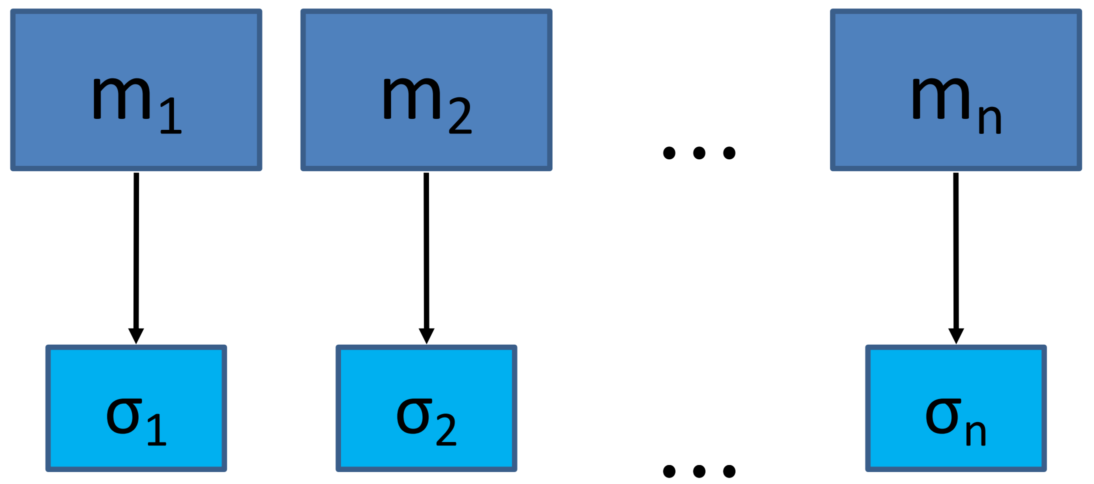
$$\sigma = \sigma_1^{\nu_1} \cdot \sigma_5^{\nu_5} \cdot \sigma_6^{\nu_6} \cdot \sigma_8^{\nu_8}$$

1. Cloud server picks a random r.
2. Computes $\lambda = e(u, g^x)^r, \gamma = h(\lambda)$
3. μ = r + γ μ mod p.

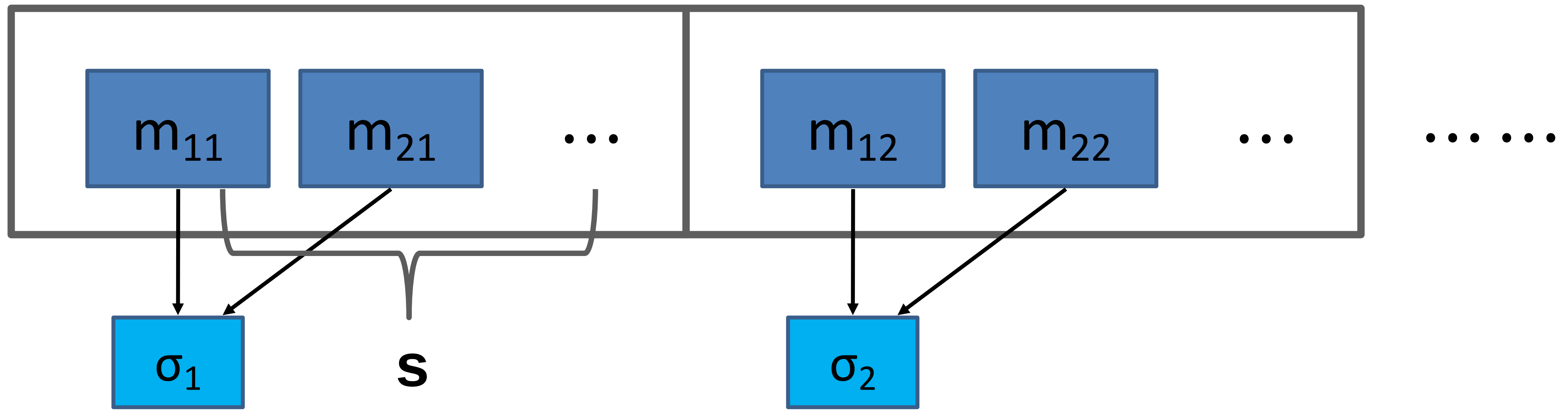
*Adopted in TC'13 (Want et al.) and many follow-ups

Storage / bandwidth tradeoff for HLA

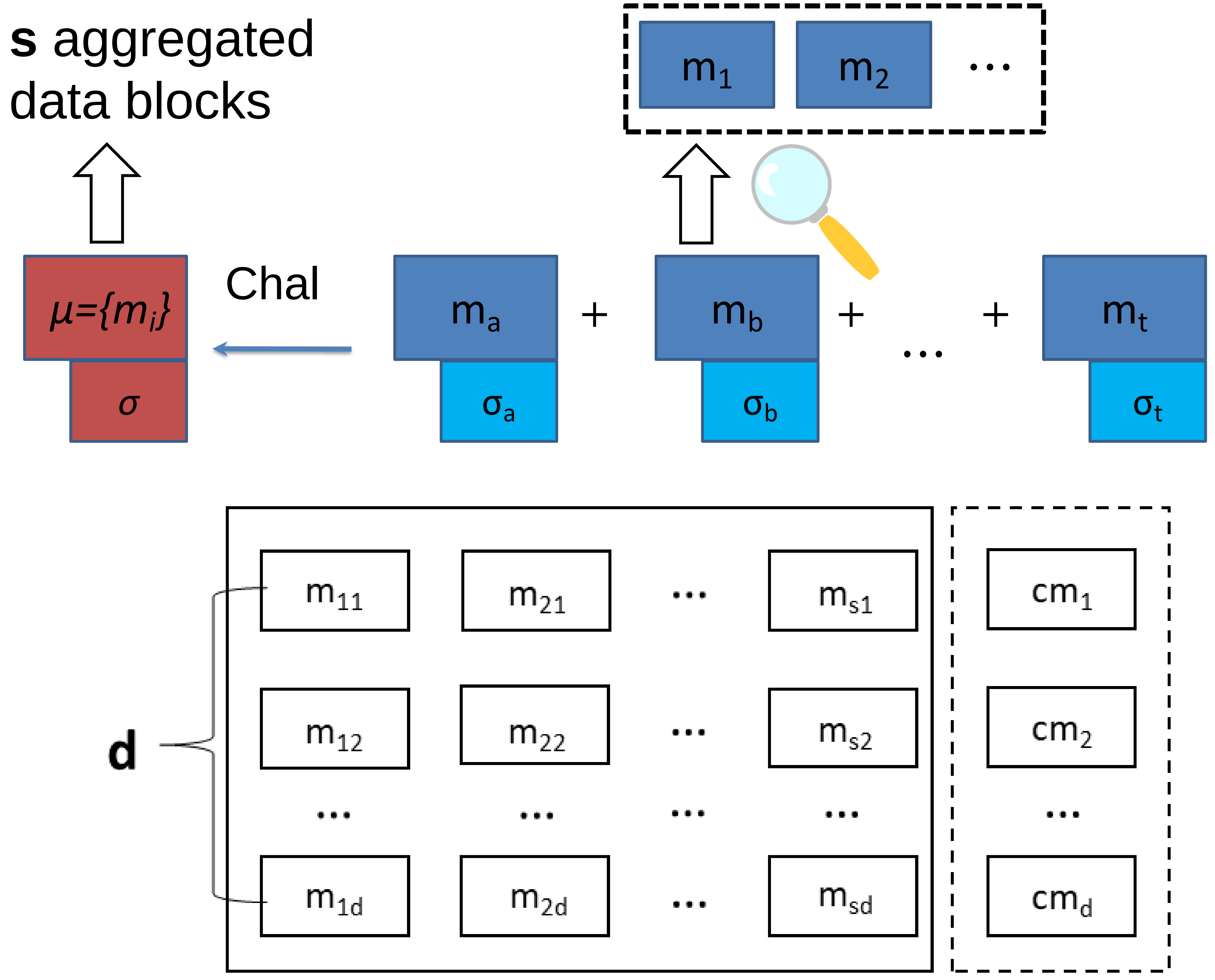
- Standard HLA has one authenticator per block
  - Per block authenticator generation can be costly
  - Authenticators would double the storage
  - Response / proof size is small



- If adopting a tradeoff parameter s
  - Bind s blocks with one authenticator
  - 1/s preprocess cost; 1/s storage overhead
  - s times response / proof size



Efficiency refinements by polynomial commitment

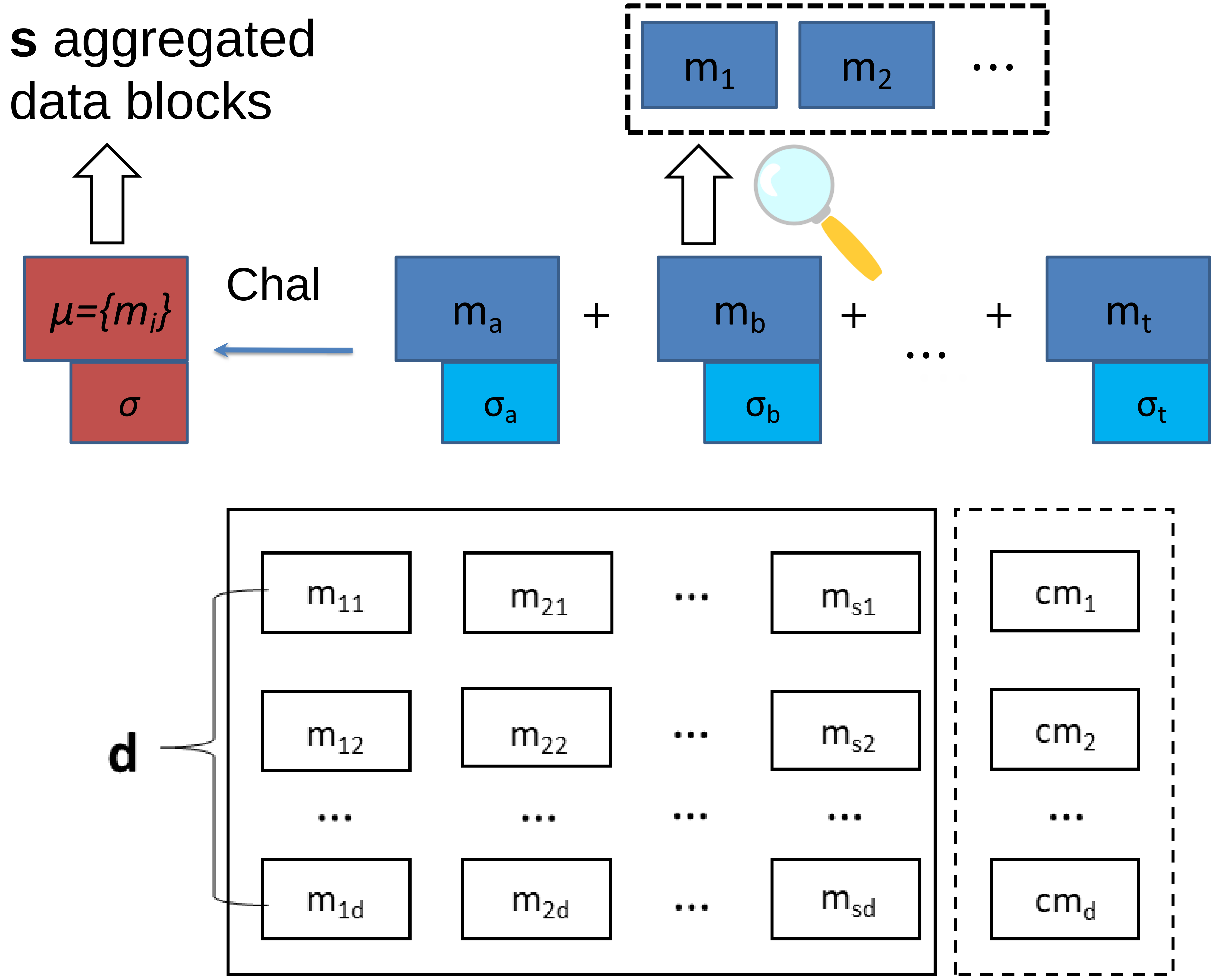


Solid line: data blocks represented as big number; Dotted line: authenticators in the form of polynomial commitments

- Increased proof size yields undesirable on-chain overhead
  - μ now expanded by s times
- Leveraging polynomial commitment[1]
  - From O(s) to O(1) proof size, same as HLA without tradeoff parameter

1. Kate., et al. "Constant-Size Commitments to Polynomials and Their Applications." AsiaCrypt'10

Efficiency refinements by polynomial commitment



Solid line: data blocks represented as big number; Dotted line: authenticators in the form of polynomial commitments

- High-level idea of polynomial commitment
  - *For any polynomial f(x) and value r, (x-r) divides the polynomial f(x)–f(r)*
  - *Prover can compute quotient polynomial*
  - *Prover can also generate commitment of quotient polynomial using public keys*
- The commitment can compactly represent a vector of **s** data blocks in a storage proof

Efficiency refinements by polynomial commitment

$$c_1 \begin{bmatrix} m_{11} \\ m_{21} \\ \cdots \\ m_{s1} \end{bmatrix} + c_2 \begin{bmatrix} m_{12} \\ m_{22} \\ \cdots \\ m_{s2} \end{bmatrix} + \cdots c_d \begin{bmatrix} m_{1d} \\ m_{2d} \\ \cdots \\ m_{sd} \end{bmatrix}$$

$$= \begin{bmatrix} \mu_1 \\ \mu_2 \\ \cdots \\ \mu_s \end{bmatrix} \Rightarrow$$

$$y' = \mathrm{a}\, P_k(r) + \mathrm{b}$$

$$Q_k(x) = \frac{P_k(x) - P_k(r)}{x - r}$$

$$\psi = g^{Q_k(\alpha)}$$

- Key setup: $\{g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{s-1}}\}$, other pk
- Data preprocessing: $\sigma_i = g^{M_i(\alpha)} H(name||i)^x$
- Challenge: $\{i, ci\}$ expanded through PRP & PRF
- Proof generation: a = H'(R), submit $\{y', \sigma, \psi, R\}$
  - **less than 300 bytes**
- Proof verification (high-level):

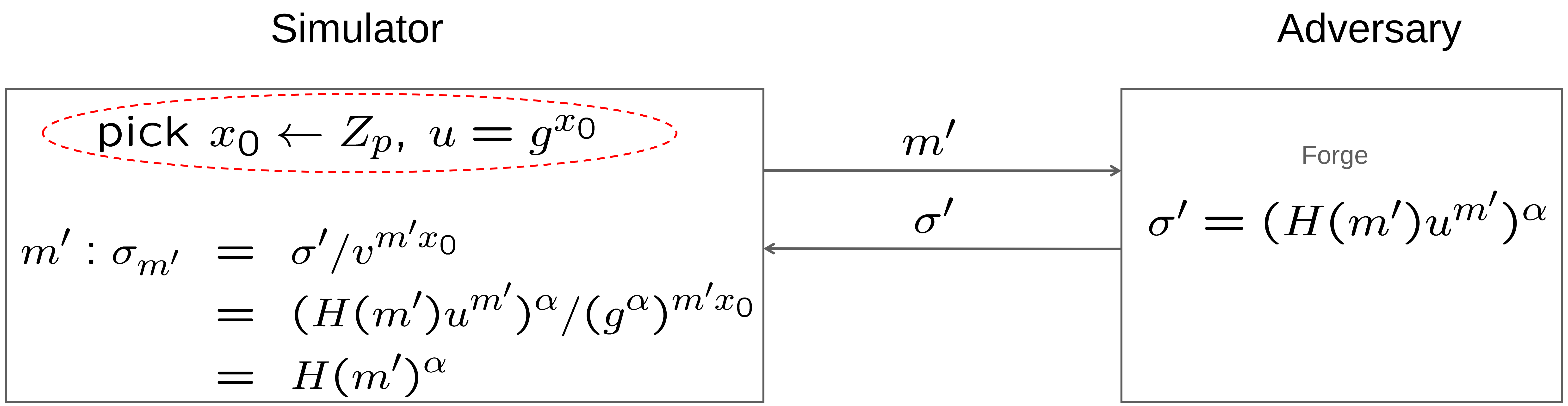
$$g^{P_k(\alpha) - P_k(r)} = g^{(\alpha - r) Q_k(\alpha)}$$

Security analysis

Our proposed design achieves the following guarantee.

- Soundness => forgeability of authenticators; Knowledge extractability;
- Probabilistic guarantee of random sampling using techniques of combinatorics
- Zero Knowledge => Witness-indistinguishable Sigma protocol
- Under the assumptions of: *Computational Diffie-Hellman (CDH)*,

  Bilinear Strong Diffie-Hellman problem (q-BSDH).

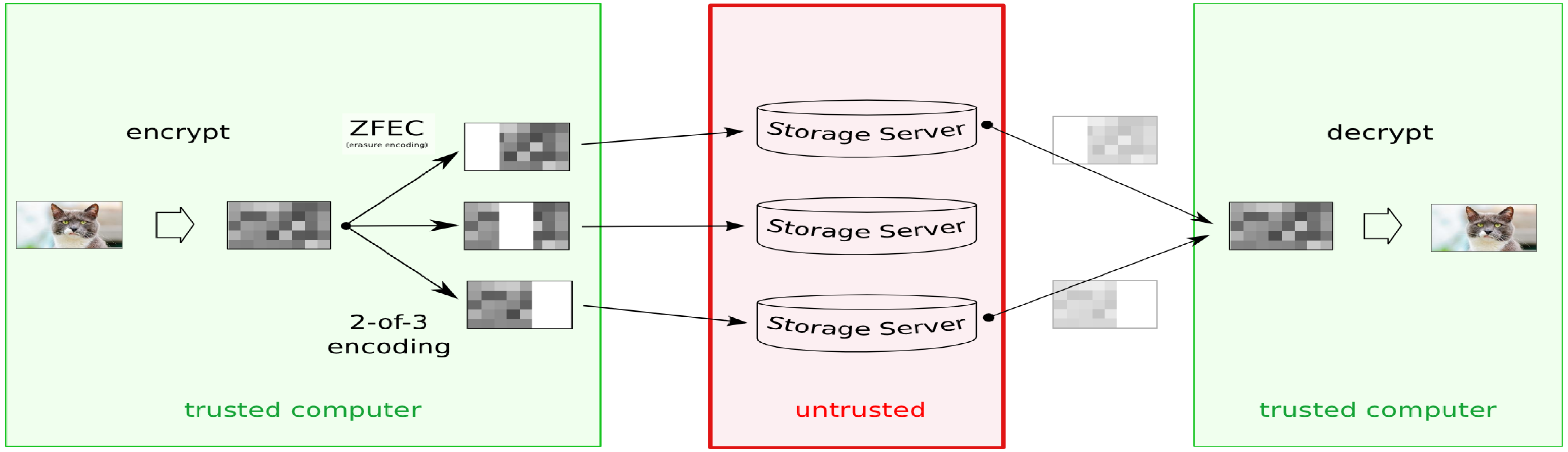


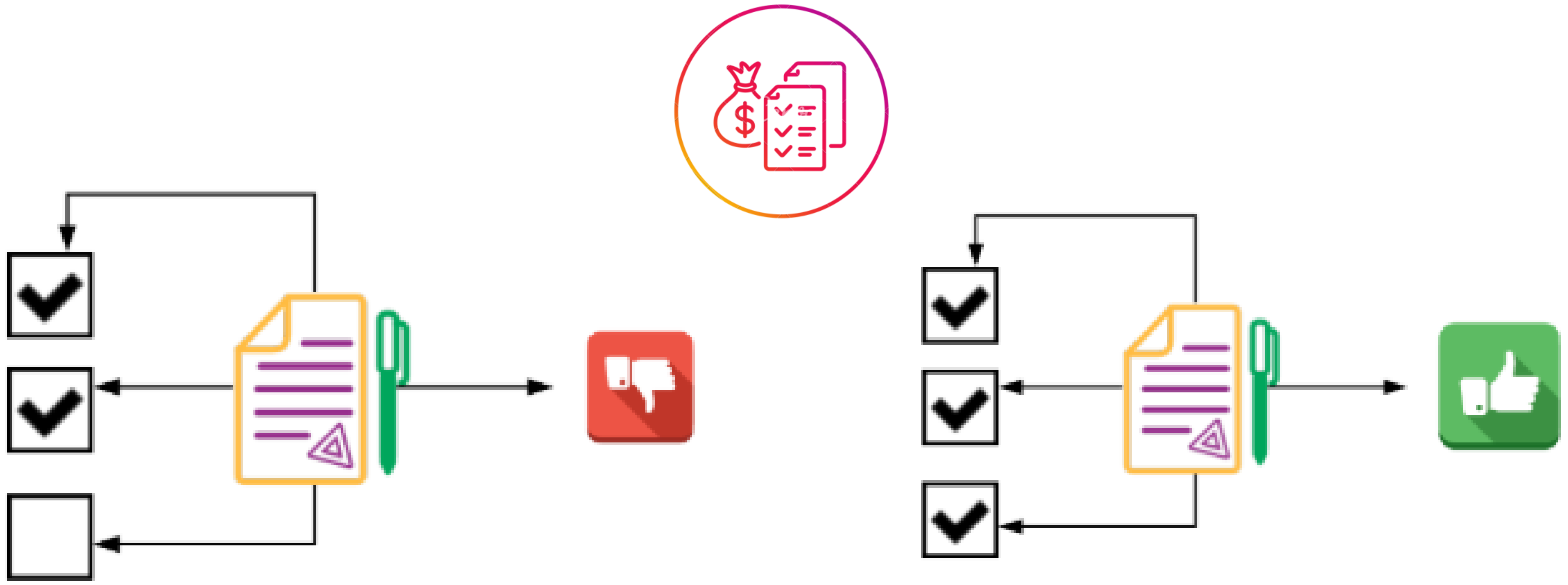
Many other practical considerations

- Generating cheap & unbiased random challenges on blockchain
- Engineering the crypto pieces together
  - e.g., limited crypto support at EVM
  - e.g., what concrete construction to use, RSA VS ECC
- …

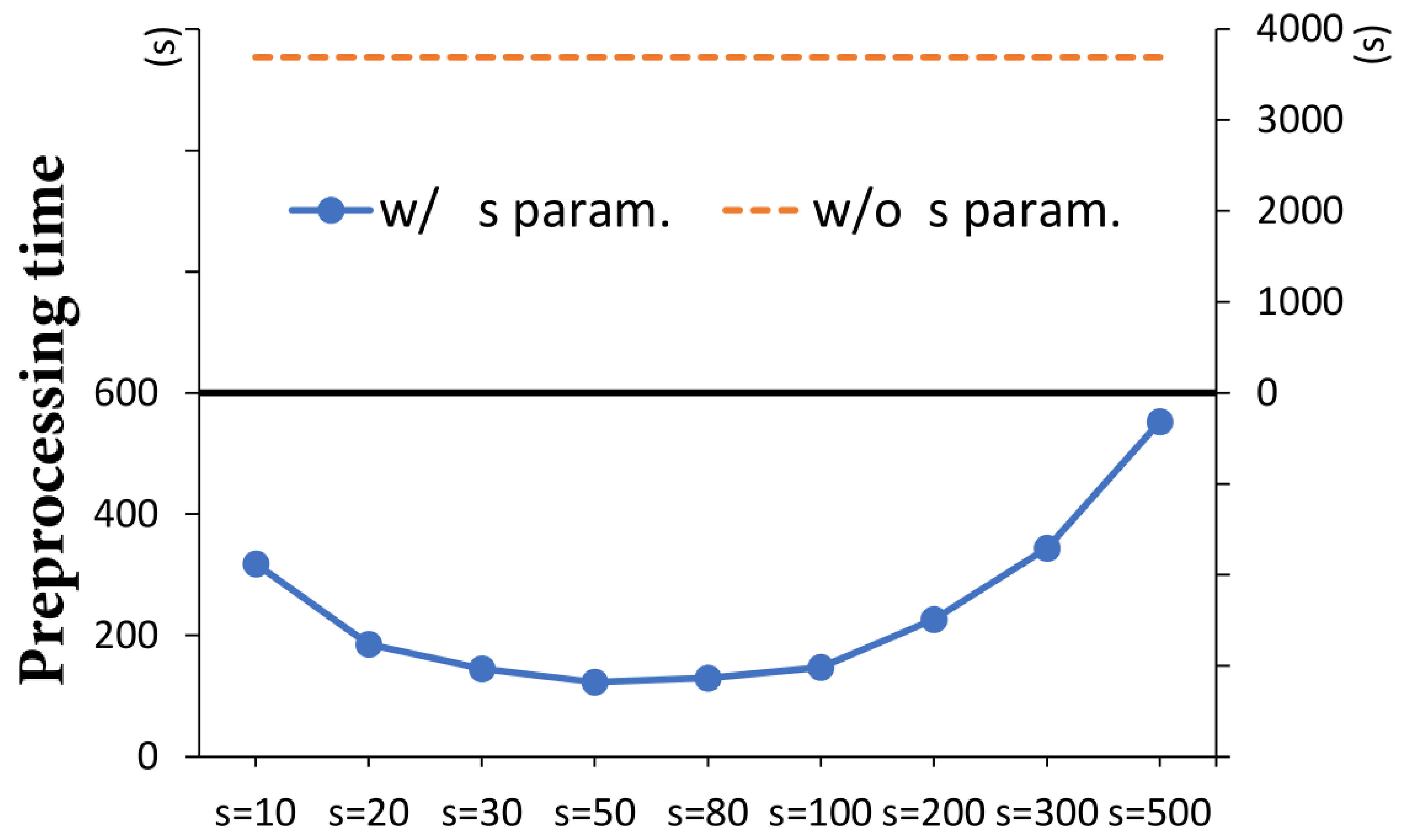
Evaluation

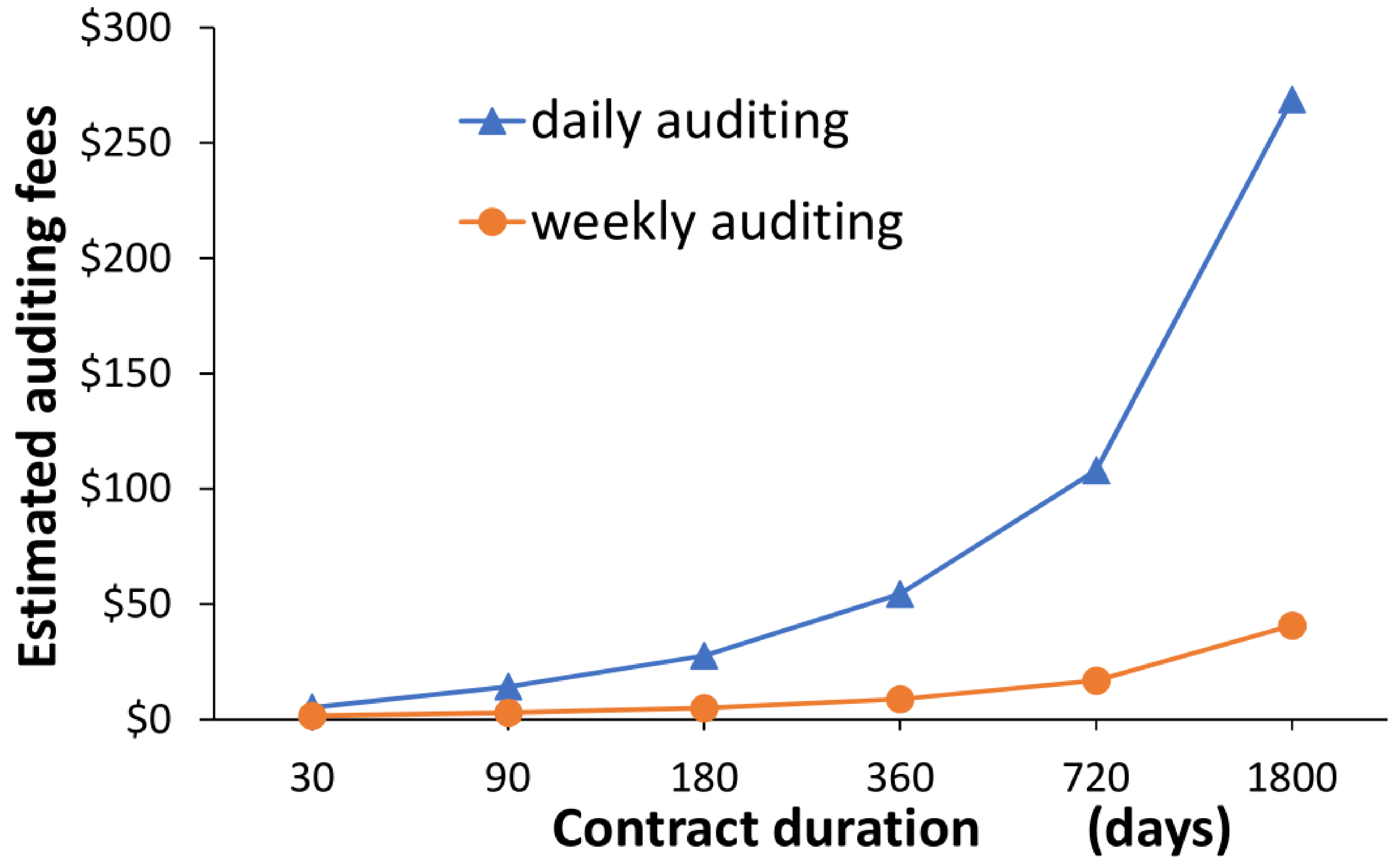
- We have developed a fully functioning prototype using the Ethereum smart contract atop of a DSN infrastructure with Tahoe-LAFS





Evaluation





- Per audit cost: **0.13 USD**
- Overall auditing fees comparable with cloud storage fees
    - If applying 3-out-of-10 coding for availability, daily auditing
- **2 min** for pre-processing a file of **1 GB** size
- Can scale to **thousands of users**
    - With adequate Blockchain throughput, batch processing on storage providers

Concluding remarks

- We propose a concrete auditing construction in the context of DSN
  - Preventing exploit of on-chain proofs
  - Concrete efficiency on both storage overhead and succinct proof size
- Our instantiation can be easily adopted to complement prior arts in continuous auditing
- Future tasks:
  - Potential support for data dynamics (possibly easier from our HLA-based direction)
  - Batching multiple proofs