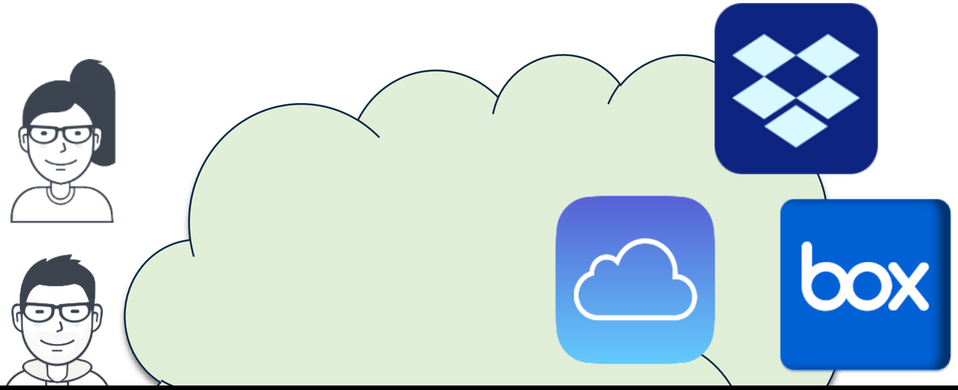
EFFICIENT DYNAMIC PROOF OF RETRIEVABILITY FOR COLD STORAGE

Tung Le, Pengzhi Huang, Attila A. Yavuz, Elaine Shi, Thang Hoang

NDSS 2023

San Diego, California

Overview



CIO EVENTS NEWSLETTERS WHITE PAPERS/WEBCASTS BRANDPOSTS COMMUNITY BLOGS PODCASTS CIO 100 AWARDS CIO DIGITAL MAGAZINE

SPONSORED

Storage as a Service: More than Just On-Demand Consumption

BrandPost Sponsored by HPE | Learn More

By Beth Joseph | JAN 18, 2022 3:38 PM PST

Economy Retirement How to Invest

...e Service ... Demands, Share, Trends, Futuristic Opportunity, Share and Forecast To 2030 By VMReports

Published: Feb. 17, 2023 at 4:39 p.m. ET

Home > Entrepreneurs > Startup Contributor

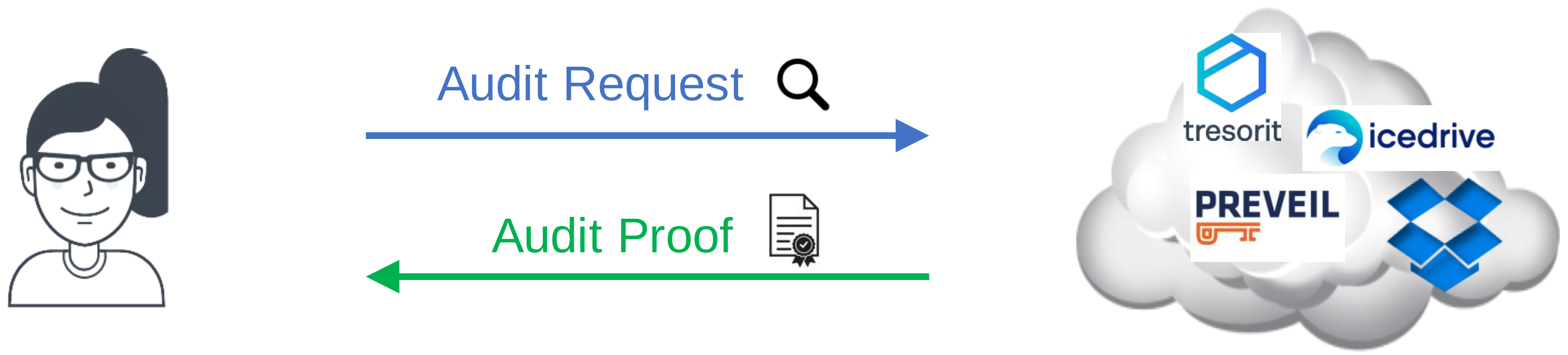
Data Explosion: The Rise of Cloud Services

By Fran Villalba Segarra - Internxt Universal Technologies SL CEO

Overview

Baseline	Static PoR (CCS' 09)
• Use HMAC to verify the integrity of data. ▪ Fast update, slow audit.	• Insert "sentinels". ▪ No update support. ▪ Limited audit times.

Our Efficient PoR Technique



Dynamic PoR (DPoR) allows efficient update ability.

Previous DPoRs	Our Work (Porla)
• Low storage (USENIX’ 21) • Fast update (CCS’ 13) • Metadata privacy (Cash’ 17)	• Small Proof Size • Low Audit Time

Error Correction Code

Error Correction Code allows recovering the entire dataset while tolerating a certain portion of damaged codewords.

github.com/vt-asaplab/porla/ICC

$$\underbrace{\mathbf{H}_\ell}_{1 \times 2^{\ell+1}} := \underbrace{\pi_\ell(\vec{v}_\ell)}_{1 \times 2^{\ell}} \times \underbrace{\boxed{[\mathbf{F}_\ell \mid \mathbf{D}_{\ell,t}\mathbf{F}_\ell]}}_{\mathbf{G}_{2^\ell \times 2^{\ell+1}}}$$

Any submatrix $2^\ell \times 2^\ell$ of $\mathbf{G}$ is non-singular

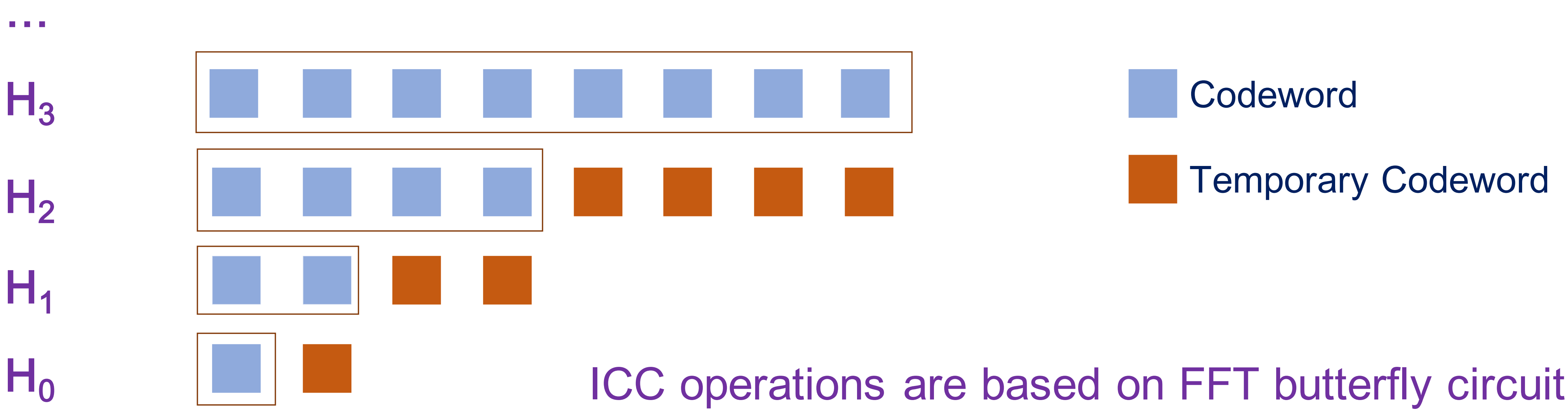
$$\pi_\ell(\vec{v}_\ell) := \underbrace{\widehat{\mathbf{H}}_\ell}_{1 \times 2^\ell} \times \underbrace{\widehat{\mathbf{G}}^{-1}}_{2^\ell \times 2^\ell}$$

Incrementally Constructible Code (ICC)

- **Raw data buffer U**.
- **Erasure code C**: ECC built from U.
- **Hierarchical log H**: Incrementally Constructible Code (ICC).

Incrementally Constructible Code (ICC)

- Level $\mathbf{H}_\ell$ is rebuilt after every 2^ℓ updates.
- After all N blocks are updated, C is rebuilt.



ICC operations are based on FFT butterfly circuit.

Homomorphic Authenticated Commitment

- **Secret key:** α
- **Data block:** $\vec{v} = (v_1, v_2, \dots, v_n)$
- **Commitment of $\vec{v}$:**

$$\mathrm{cm} := \vec{g}^{\vec{v}} \qquad \vec{g} \in \mathbb{G}^n$$

- **MAC of Commitment:**

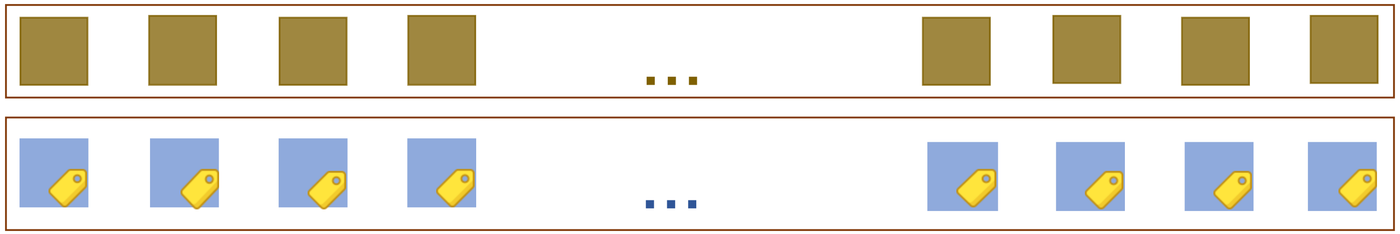
$\mathrm{PRF}_k(time, level, index)$

$$\sigma := (\vec{g}^{\vec{v}})^{\alpha} h^{r} \qquad h \in \mathbb{G}$$

MAC

Data Structures

- Raw buffer U:
- Erasure code C:



- Hierarchical Log H:

...

H_3

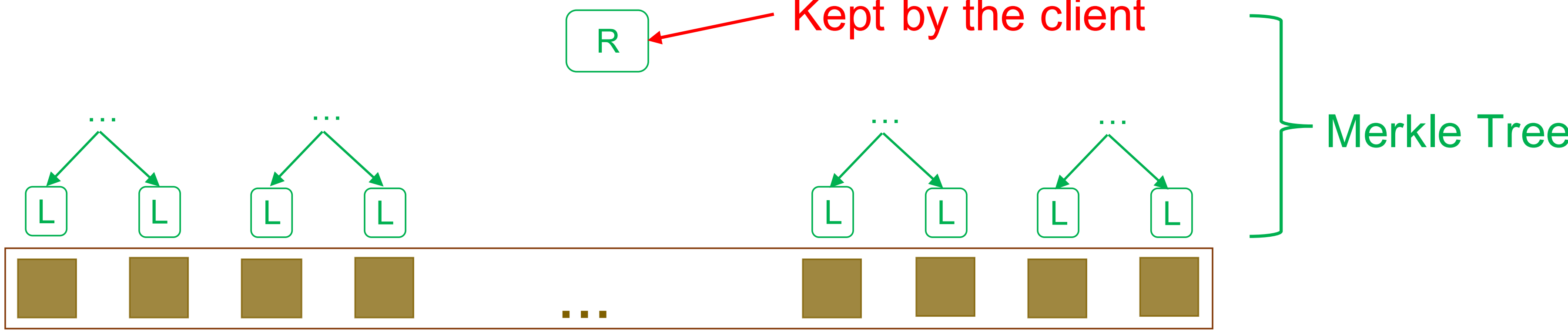
H_2

H_1

H_0

Update Procedure

- Raw buffer U:

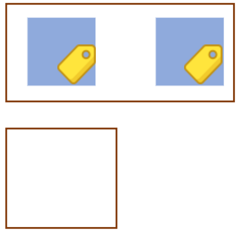


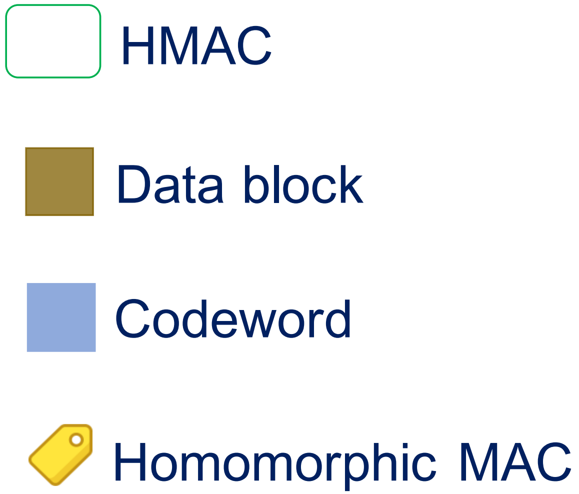
- Hierarchical Log H:

...

H_1

H_0





- Erasure code C: computed from U after N updates happen.

Audit Procedure

- Erasure code C:
- Hierarchical Log H:

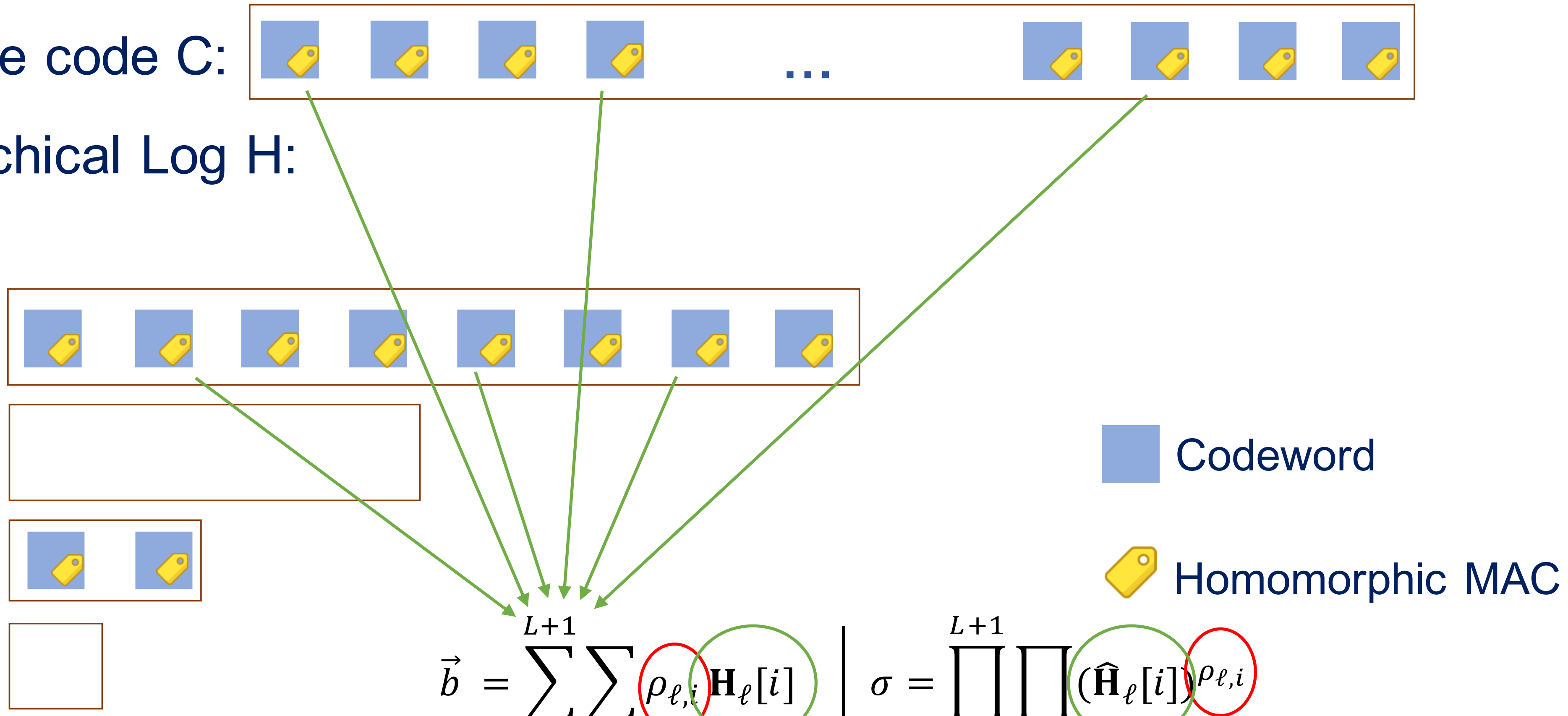
…

H_3

H_2

H_1

H_0



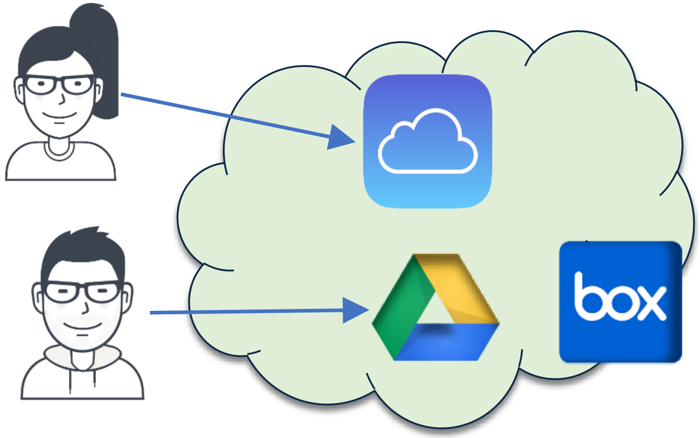
$$\vec{b} = \sum_{\ell=0}^{L+1}\sum_{i\in I}\rho_{\ell,i}\mathbf{H}_{\ell}[i] \quad \Big| \quad \sigma = \prod_{\ell=0}^{L+1}\prod_{i\in I}(\hat{\mathbf{H}}_{\ell}[i])^{\rho_{\ell,i}}$$

random scalars indicated by the client

λ random codewords/MACs on C and each $\mathbf{H}_\ell$

A random linear combination

Our Audit Protocol



$$\vec{b} = \sum_{\ell=0}^{L+1}\sum_{i\in I}\rho_{\ell,i}\,\mathbf{H}_\ell[i] \quad \Big| \quad \sigma = \prod_{\ell=0}^{L+1}\prod_{i\in I}(\hat{\mathbf{H}}_\ell[i])^{\rho_{\ell,i}}$$

α: secret key

Audit request + a random seed

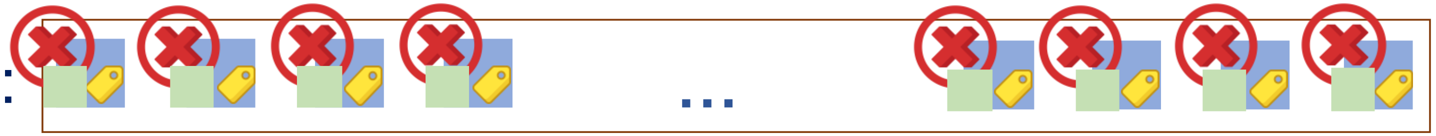
Commitment & MAC of aggregated codeword block

$\mathrm{cm}_{\vec{b}};\sigma$

$$(\mathrm{cm}_{\vec{b}})^{\alpha}\prod_{\ell=0}^{L+1}\prod_{i\in I}(h^{r(t,\ell,i)})^{\rho_{\ell,i}} \stackrel{?}{=} \sigma$$

Not Enough!

- Erasure code C:



- Hierarchical Log H:

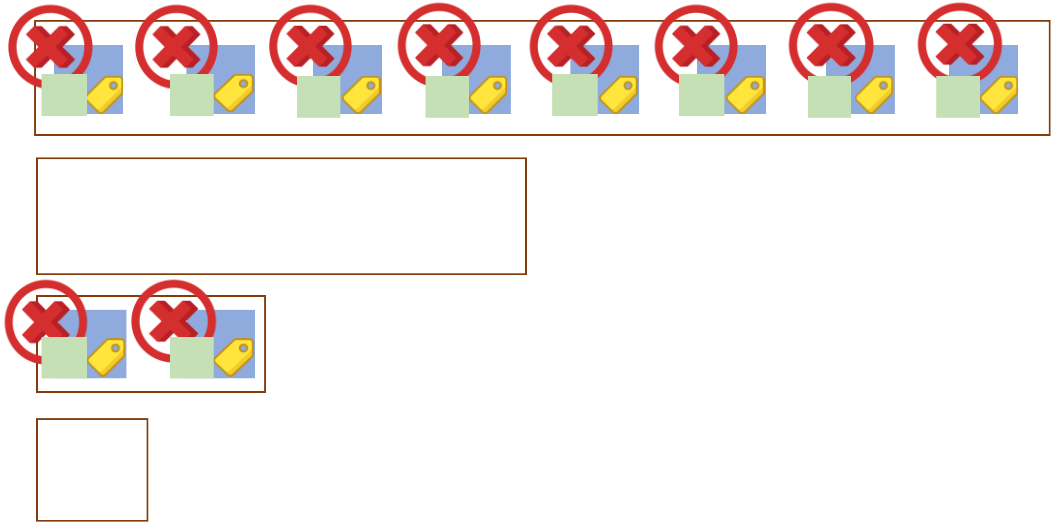
…

H_3

H_2

H_1

H_0



Codeword

Homomorphic MAC

Commitment

Our Audit Protocol

α: secret key

Audit request + a random seed

Commitment & MAC of aggregated codeword block

+ Proof of aggregated codeword block

$\mathrm{cm}_{\vec{b}}; \sigma; \pi$

- Verify proof π
- $(\mathrm{cm}_{\vec{b}})^{\alpha} \prod_{\ell=0}^{L+1} \prod_{i \in I} (h^{r(t,\ell,i)})^{\rho_{\ell,i}} \stackrel{?}{=} \sigma$

If π is $\vec{b}$, then the total audit proof size is large if size of $\vec{b}$ is large

Bulletproofs (IEEE S&P' 18)	KZG (AsiaCrypt' 10)
• Not require trusted setup. • $\lvert\pi\rvert = \mathcal{O}(\log D)$	• Required trusted setup. • $\lvert\pi\rvert = \mathcal{O}(1)$

Proof of Polynomial Evaluation

Proof of the server's knowledge of the aggregated codeword given its commitment P.

Scheme	Porla$_{\text{ipa}}$	Porla$_{\text{kzg}}$
Audit Proof Size	$(2\log(D)+2)\lvert\mathbb{G}\rvert + 2\lvert\mathbb{Z}_p\rvert$	$3\lvert\mathbb{G}\rvert$

aggregated codeword $\vec{x} = (x^0, x^1, \dots x^{D-1})$

$\vec{b}, \vec{x}$: $P = \vec{g}^{\vec{b}}$ and $c = \langle \vec{b}, \vec{x} \rangle$

$\vec{x}$

An evaluation point

x

π

Proof

$\vec{g} \in \mathbb{G}^D$: independent generators

$\vec{b} \in \mathbb{Z}_p^D, c \in \mathbb{Z}_p, P \in \mathbb{G}$

Evaluation - Configuration

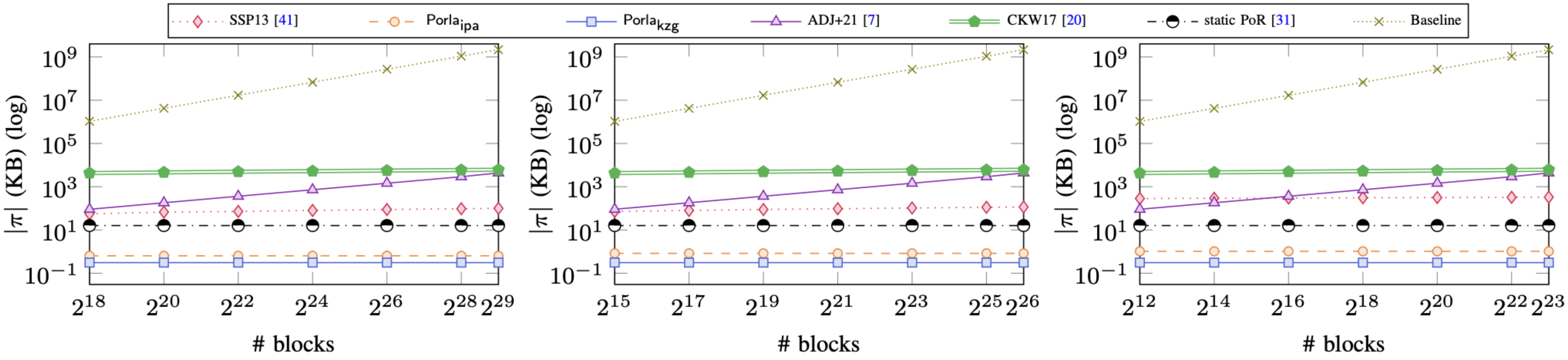
- **Server:**
  - Amazon EC2 c6i.8xlarge.
  - 16-core Intel Xeon Platinum 8375C CPU @ 2.9 GHz.
  - 64 GB RAM.

- **Client**:
  - Intel i7-6820HQ CPU @ 2.7 GHz.
  - 16 GB RAM.

- **Implementation**:
  - C++ with ~4,000 LOCs.
  - Secp256k1 ($Porla_{ipa}$), BN254 ($Porla_{kzg}$)

Evaluation – Audit Proof Size

$87\times - 14{,}012\times$ smaller proof size than previous DPoR schemes.



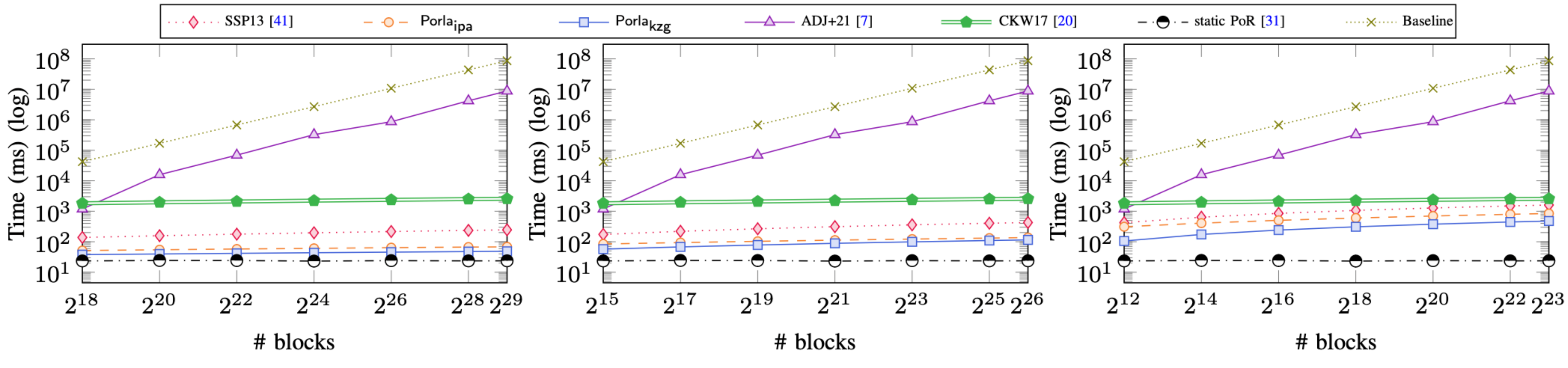
(a) $\beta = 4\,\text{KB}$ **(b)** $\beta = 32\,\text{KB}$ **(c)** $\beta = 256\,\text{KB}$

Scheme	Proof Size (KB)
$\text{Porla}_{\text{ipa}}$	0.64 - 1.03
$\text{Porla}_{\text{kzg}}$	0.31

Independent with database size

Evaluation – Audit Delay

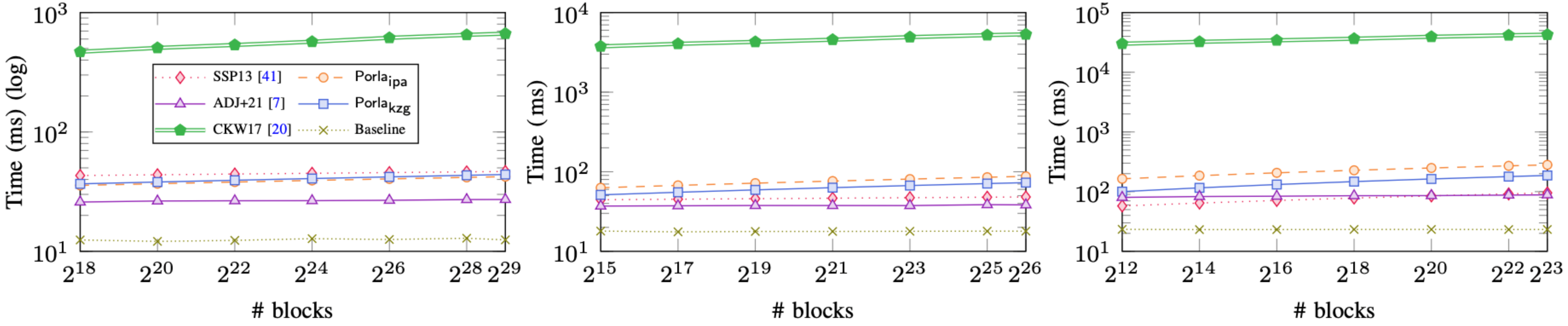
$4\times - 18{,}000\times$ faster audit time than prior approaches.



(a) $\beta = 4$ KB (b) $\beta = 32$ KB (c) $\beta = 256$ KB

Block Size	Porla$_{ipa}$ (ms)	Porla$_{kzg}$ (ms)
4 KB	51.52 – 68.04	37.77 – 48.66
32 KB	84.42 – 137.44	57.26 – 114.98
256 KB	310.61 – 843.84	105.85 – 478.68

Evaluation – Update Latency



(a) $|B| = 4\,\text{KB}$ (b) $|B| = 32\,\text{KB}$ (c) $|B| = 256\,\text{KB}$

$1.2\times - 3\times$ slower update than the counterpart using the same ECC (Shi, CCS' 13).

Public Audit

- Erasure code C:
- Hierarchical Log H:

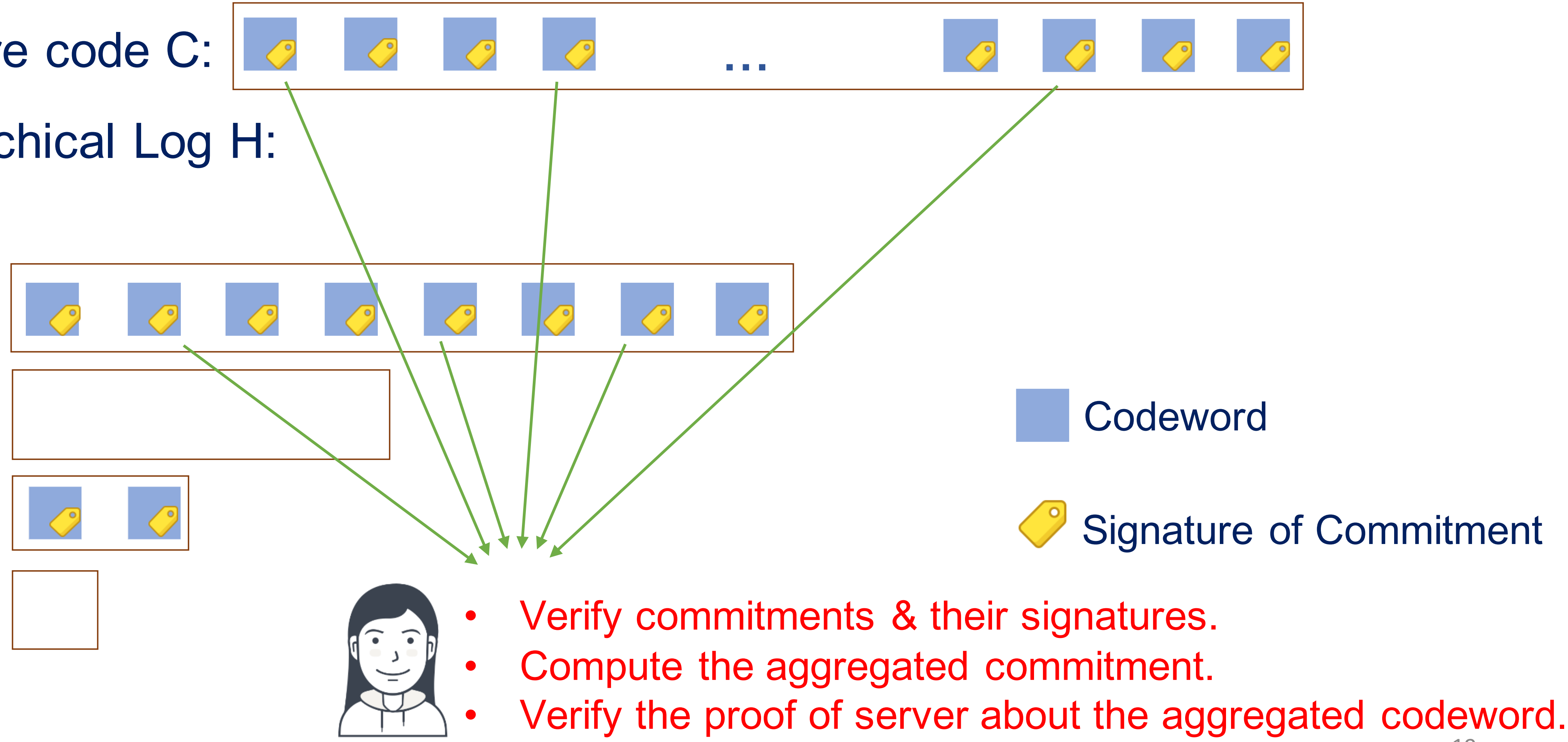
…

H_3

H_2

H_1

H_0



Conclusion & Future Work

Our Porla:

- Small audit cost: proof size and end-to-end latency.
- Maintain a reasonable data update performance.

Our source code is available at: github.com/vt-asaplab/porla

Thank you for your attention

Q&A