

Cornell University®



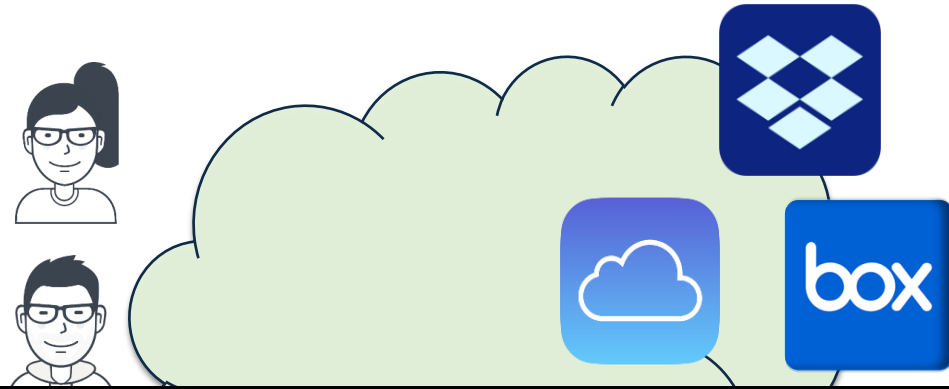
Carnegie
Mellon
University

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



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](#)

Home > Entrepreneurs > Startup Contributor

Data Explosion: The Rise of Cloud Services

By Fran Villalba Segarra - Internxt Universal Technologies SL CEO

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

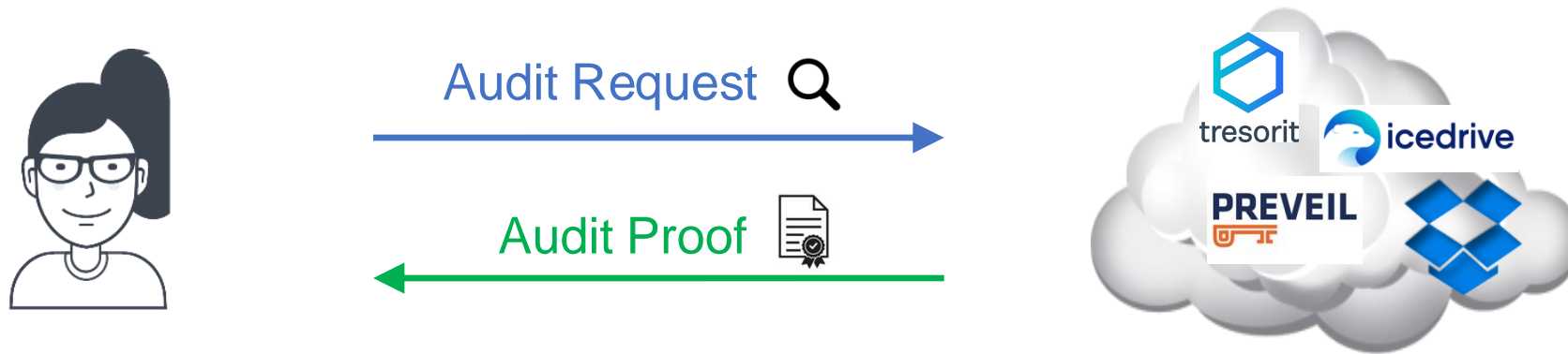
Published: Feb. 17, 2023 at 4:39 p.m. ET

Overview



Baseline	Static PoR (CCS' 09)
<ul style="list-style-type: none">• Use HMAC to verify the integrity of data.▪ Fast update, slow audit.	<ul style="list-style-type: none">• Insert “sentinels”.▪ No update support.▪ Limited audit times.

Our Efficient PoR Technique



Dynamic PoR (DPoR) allows efficient update ability.

Previous DPoRs	Our Work (Porla)
<ul style="list-style-type: none">• Low storage (USENIX' 21)• Fast update (CCS' 13)• Metadata privacy (JoC' 17)	<ul style="list-style-type: none">• 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

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

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

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

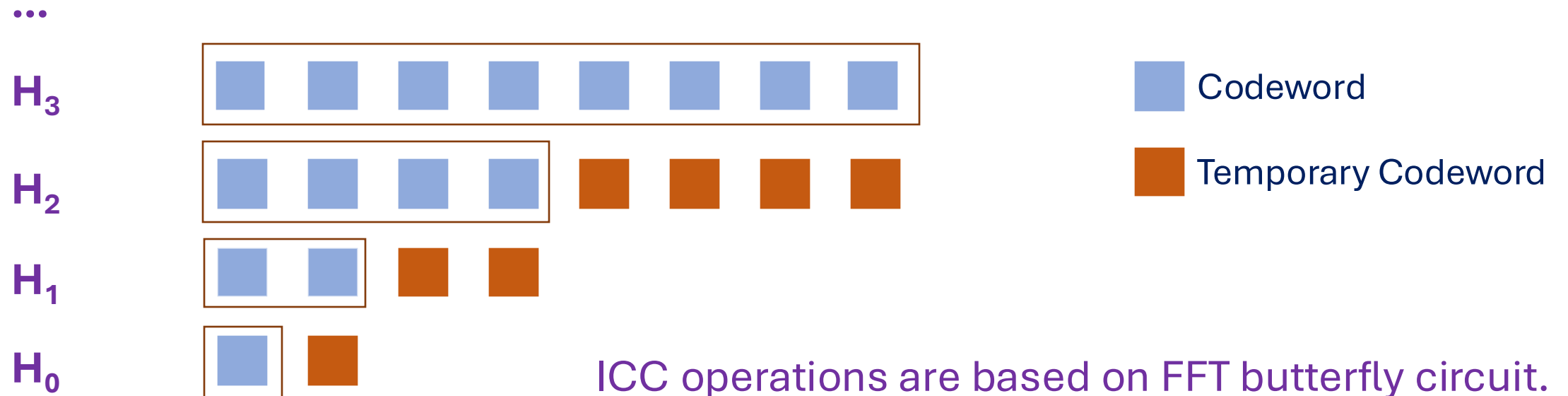
Incrementally Constructible Code (ICC)

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



Incrementally Constructible Code (ICC)

- Level H_ℓ is rebuilt after every 2^ℓ updates.
- After all N blocks are updated, C is rebuilt.



Homomorphic Authenticated Commitment



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

$$\text{cm} := \vec{g}^{\vec{v}} \quad \vec{g} \in \mathbb{G}^n$$

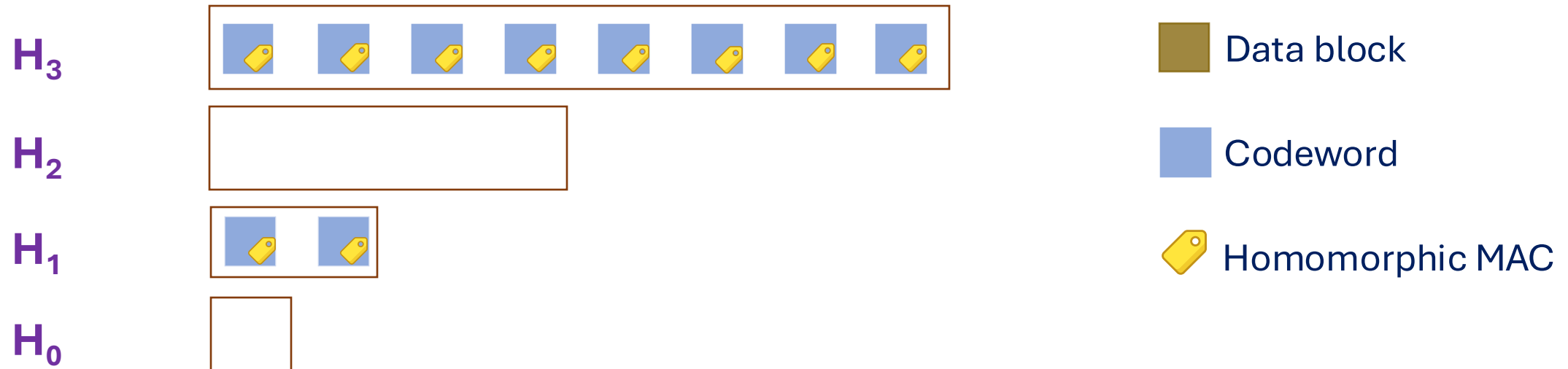
- **MAC of Commitment:**

$$\sigma := (\vec{g}^{\vec{v}})^{\alpha} \underbrace{h^r}_{\text{MAC}} \quad \text{PRF}_k(\text{time, level, index}) \quad h \in \mathbb{G}$$

Data Structures

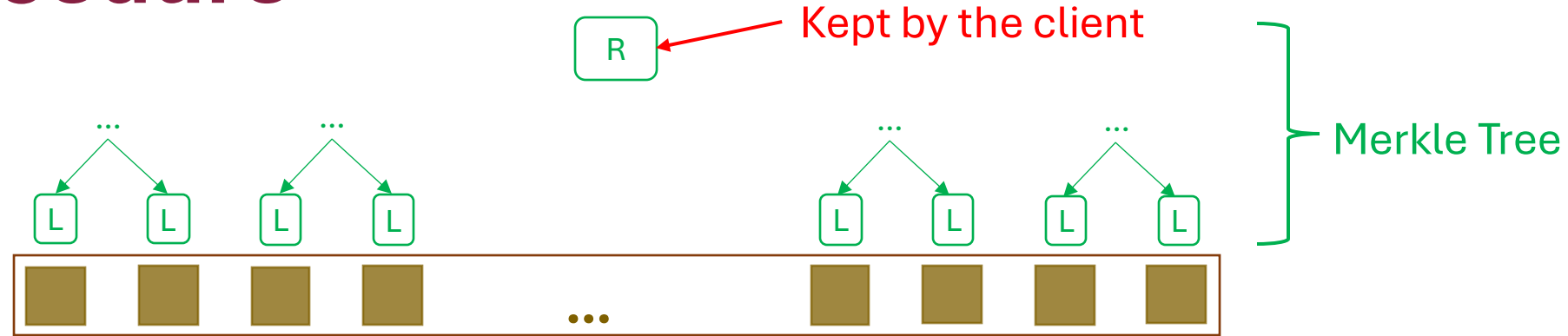
- Raw buffer U: 
- Erasure code C: 
- Hierarchical Log H:

...



Update Procedure

- Raw buffer U:

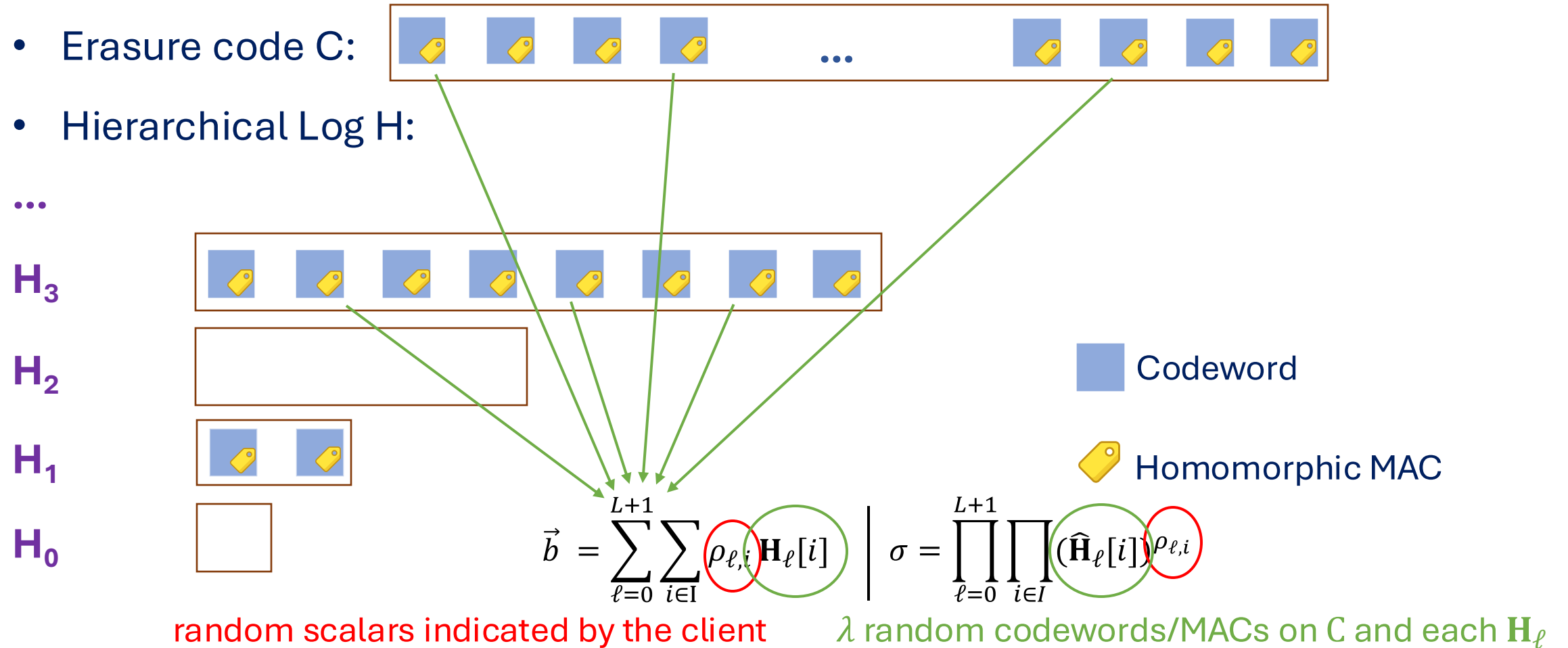


- Hierarchical Log H:



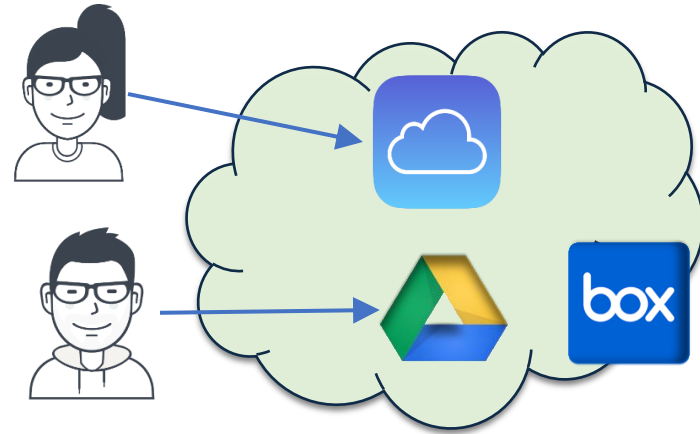
- Erasure code C: computed from U after N updates happen.

Audit Procedure



A random linear combination

Our Audit Protocol



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

α : secret key



Audit request + a random seed

Commitment & MAC of aggregated codeword block

$cm_{\vec{b}}; \sigma$



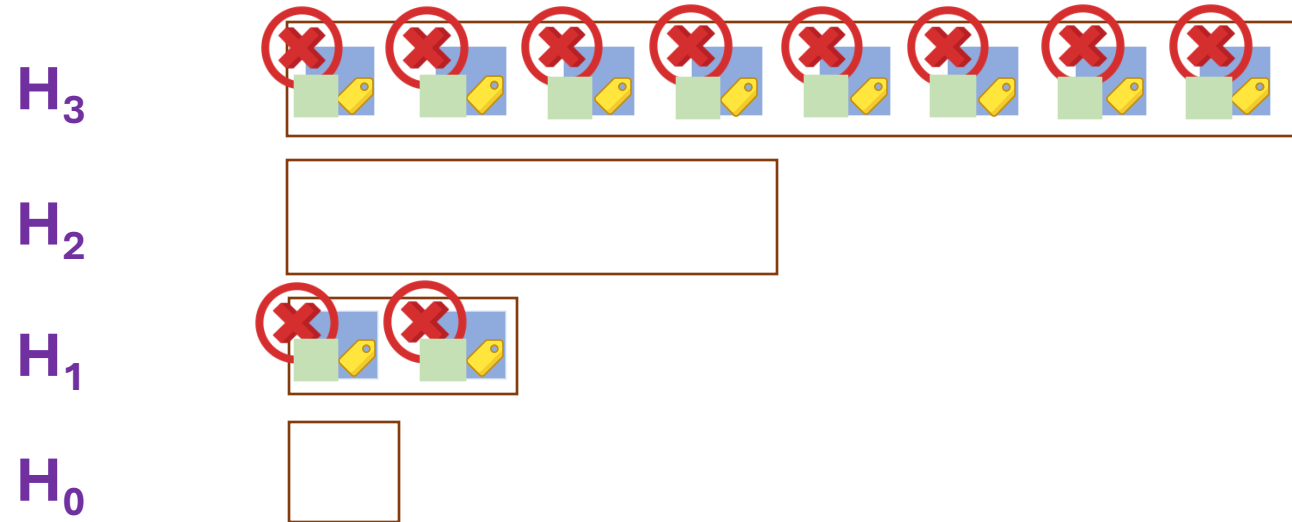
$$(cm_{\vec{b}})^{\alpha} \prod_{\ell=0}^{L+1} \prod_{i \in I} (h^{r(t, \ell, i)})^{\rho_{\ell,i}} \stackrel{?}{=} \sigma$$




Not Enough!

- Erasure code C: 

- Hierarchical Log H:

...



-  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

$cm_{\vec{b}}; \sigma; \pi$



- Verify proof π

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

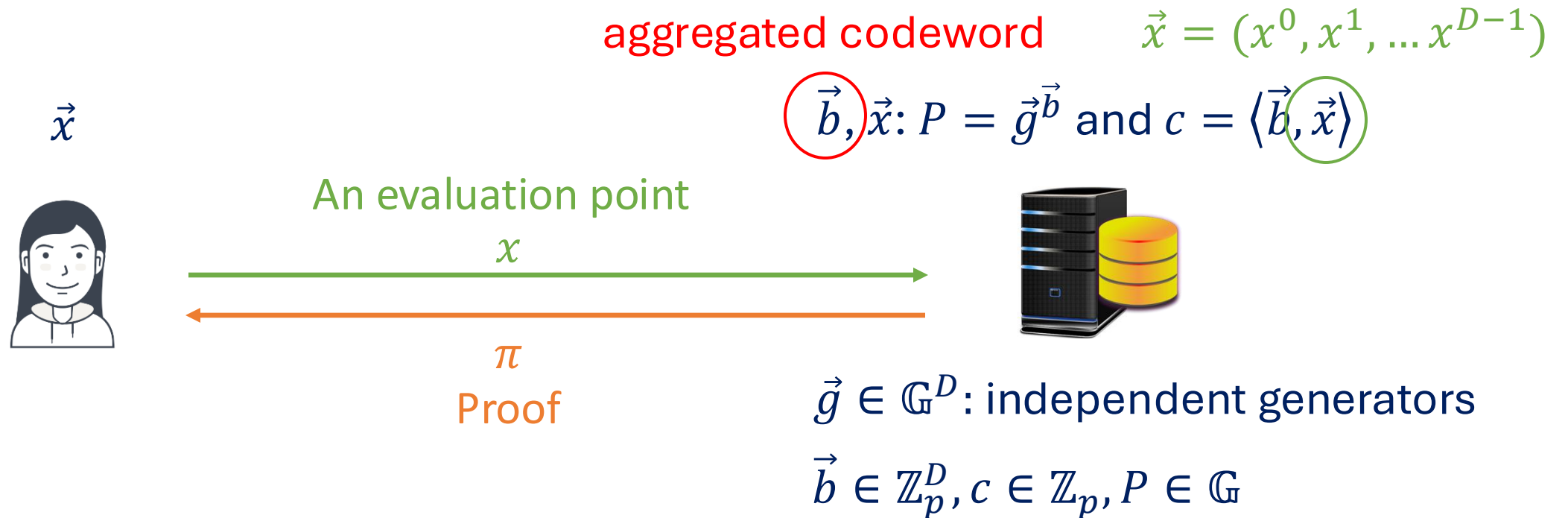
If π is \vec{b} , then the total audit proof size is large if size of \vec{b} is large

Bulletproofs (IEEE S&P' 18)	KZG (AsiaCrypt' 10)
<ul style="list-style-type: none"> • Not require trusted setup. • $\pi = \mathcal{O}(\log D)$ 	<ul style="list-style-type: none"> • Required trusted setup. • $\pi = \mathcal{O}(1)$

Proof of Polynomial Evaluation

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

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

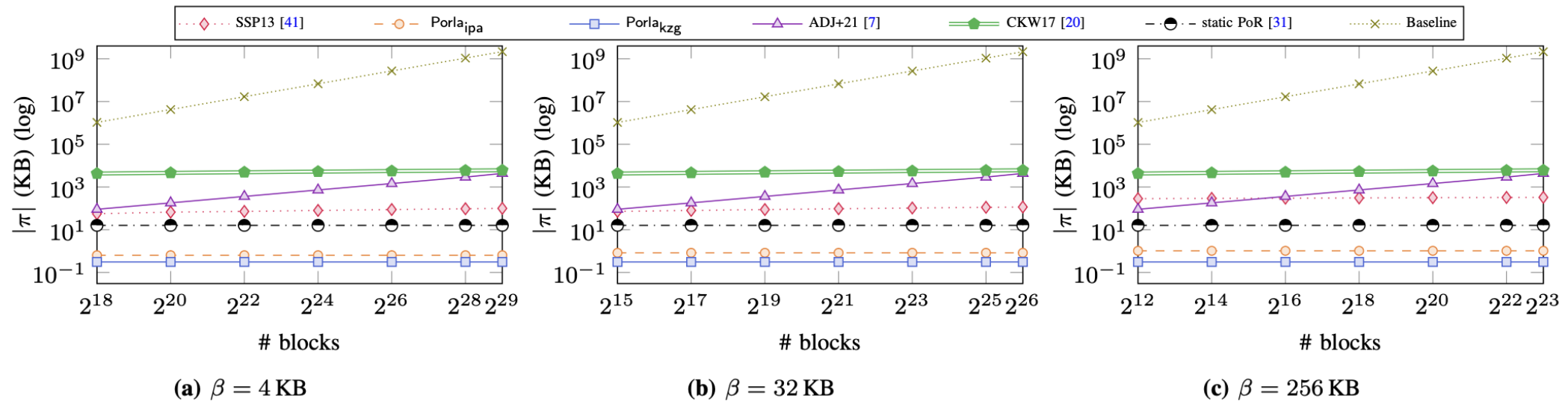


Evaluation - Configuration

- **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 × –14,012 × smaller proof size than previous DPoR schemes.

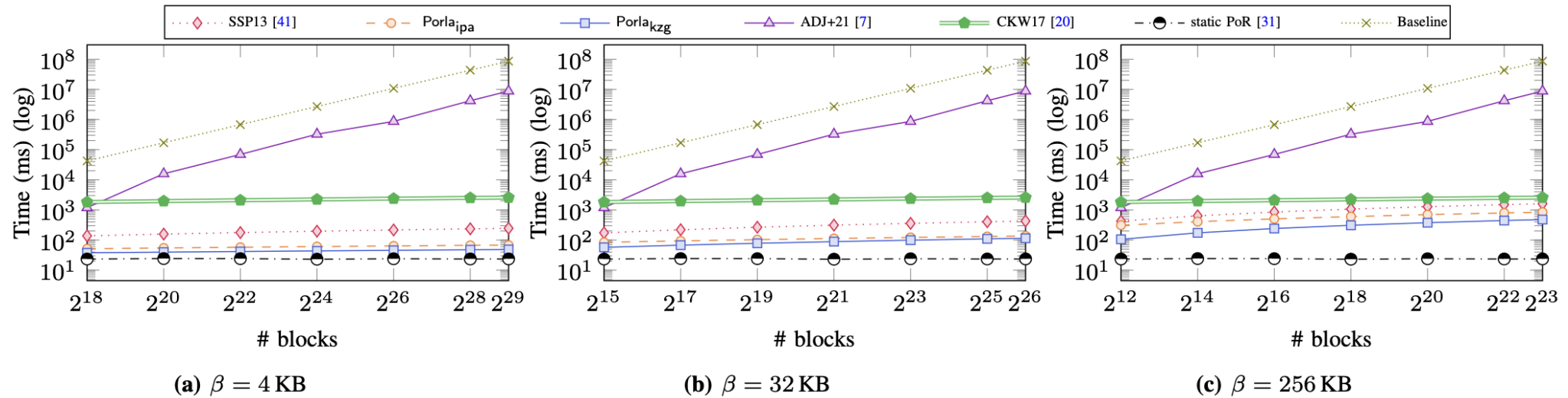


Scheme	Proof Size (KB)
Porla _{ipa}	0.64 - 1.03
Porla _{kzg}	0.31

➡ Independent with database size

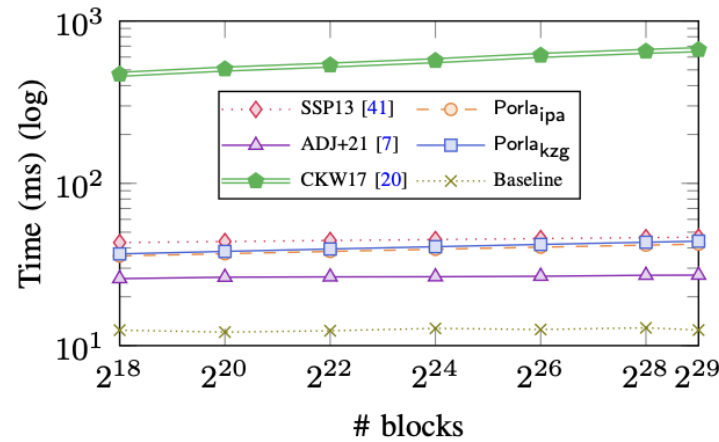
Evaluation – Audit Delay

4 × –18,000 × faster audit time than prior approaches.

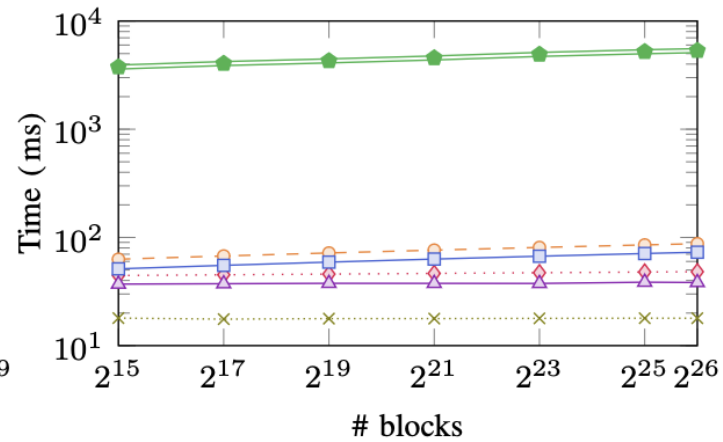


Block Size	Porla _{ippa} (ms)	Porla _{kzg} (ms)
4 KB	51.52 – 68.04	37.77 – 48.66
32 KB	84.42 – 137.44	57.26 – 114.98
256 KB	310.61 – 843.84	105.85 – 478.68

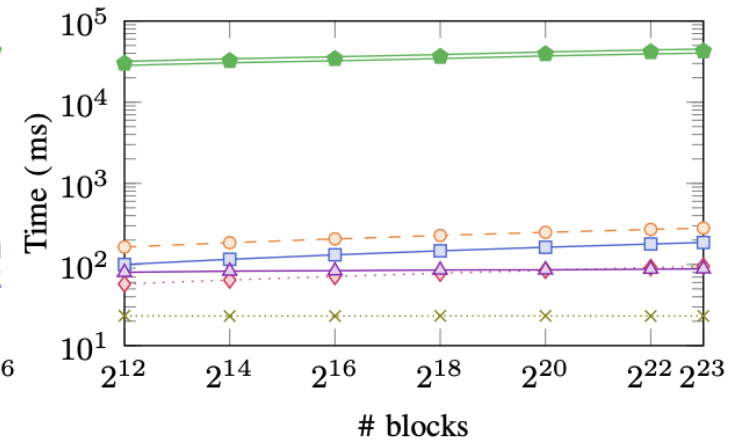
Evaluation – Update Latency



(a) $|B| = 4$ KB



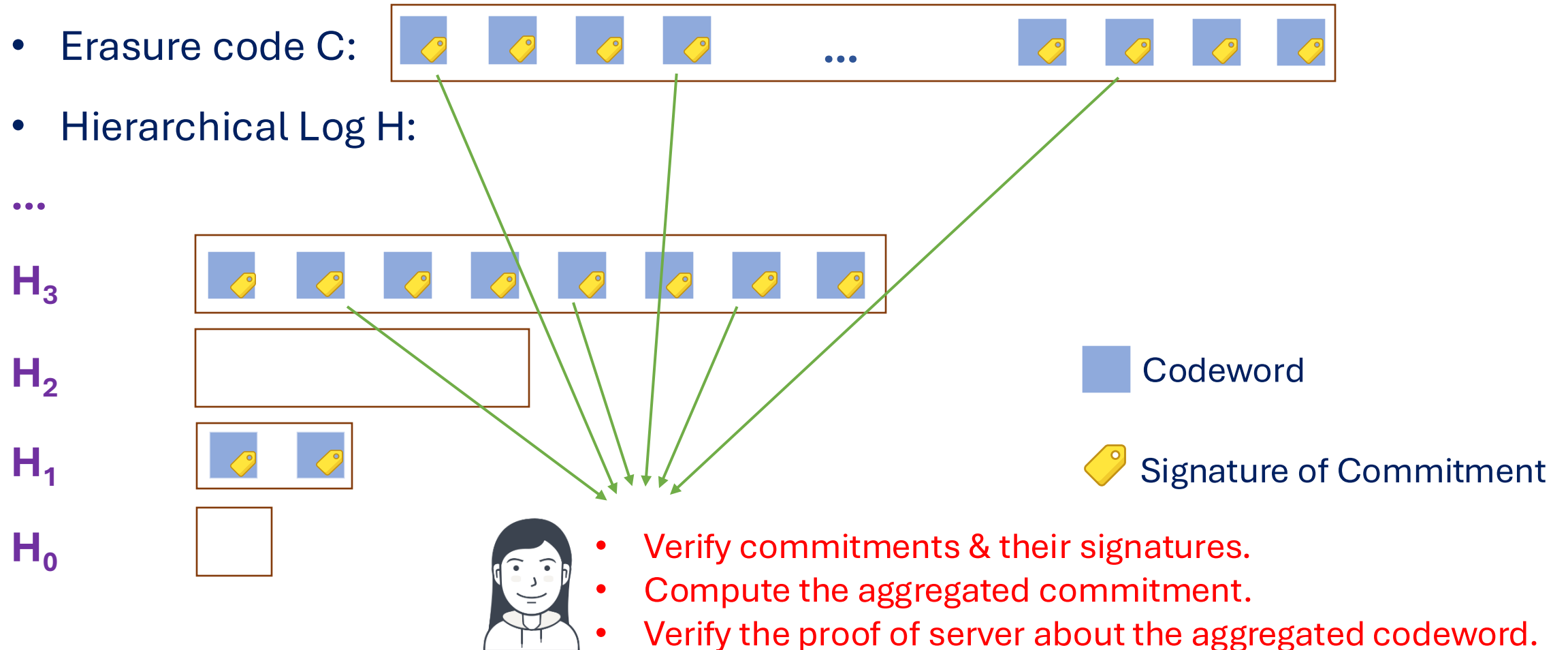
(b) $|B| = 32$ KB



(c) $|B| = 256$ KB

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

Public Audit



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