libVESPo, a library for the Verified Evaluation of Secret Polynomials

& Dynamic proofs of retrievability

Jean-Guillaume Dumas[1] Aude Maignan[1] Clément Pernet[1] Daniel S. Roche[2]

[1] Laboratoire Jean Kuntzmann
Université Grenoble Alpes
France

[2] Computer Science Department
United States Naval Academy
Annapolis, Maryland, U.S.A.

Université Grenoble Alpes

RTCA: Mathematical Software and High Performance Algebraic Computing, Lyon, Jun. 2023

Outline

Outline

1. Dynamic Proof of Retreivability
   - State-of-the-art
   - Lower bound
2. Probabilistic Verifiable Computation strategy
3. Verified evaluation of secret polynomials
4. Public auditing
5. Conclusion

Dynamic Proof of Retreivability

The Problem

- Ensure the integrity of remotely-stored data

Challenges

⇨ Want efficient reads, updates, and audits

⇨ Prior solutions either don't check everything (**incomplete**) or require replicated and encrypted storage (**non-transparent**)

Dynamic Proof of Retreivability

The Problem

- Ensure the integrity of remotely-stored data

Challenges

⇨ Want efficient reads, updates, and audits

⇨ Prior solutions either don't check everything (**incomplete**) or require replicated and encrypted storage (**non-transparent**)

Our Work

✓ Lower bound: inherent (audit time / complete check / replicated storage) tradeoff

✓ New solution: complete checks and transparent storage, but linear-time server cost for audits

✓ Privately-verifiable and publicly-verifiable versions

✓ Experiments show audits are actually fairly fast and cheap on commercial cloud

Characters in the Story

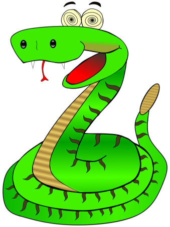
Client

Honest, but **limited** brains and memory

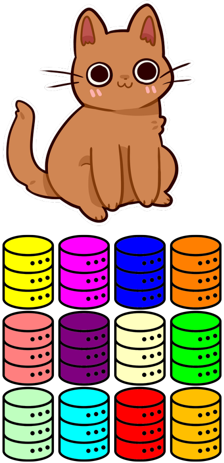
Characters in the Story

Client
Honest, but **limited** brains and memory

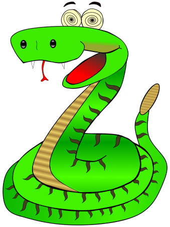
Server
Powerful but sneaky; **not to be trusted**



Characters in the Story



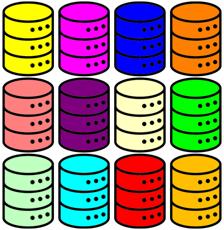
Client
Honest, but **limited** brains and memory



Server
Powerful but sneaky; **not to be trusted**

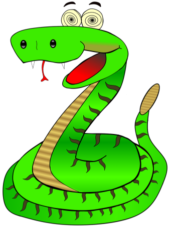
Data
Owned by client, stored on server
Could be any byte stream (not necessarily an image)

Characters in the Story



Client

Honest, but **limited** brains and memory



Server

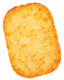
Powerful but sneaky; **not to be trusted**

Data

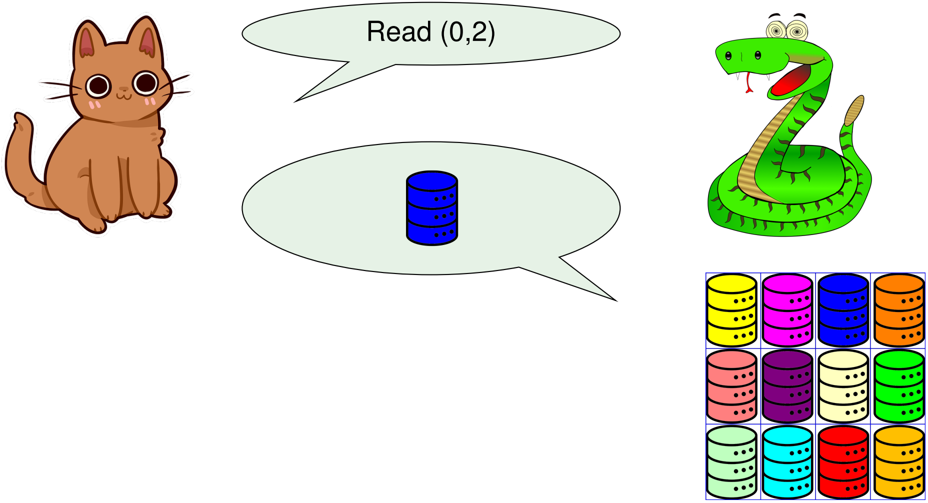
Owned by client, stored on server

Could be any byte stream (not necessarily an image)

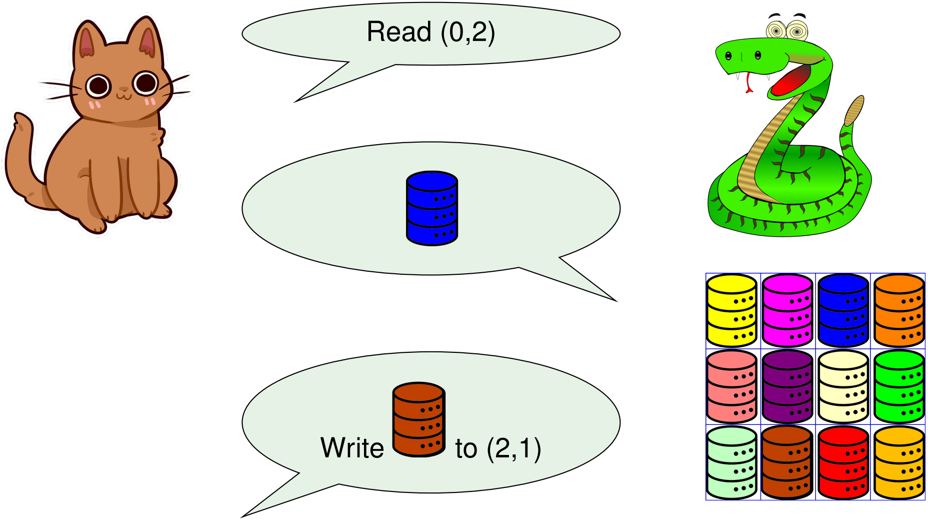
Hash digest



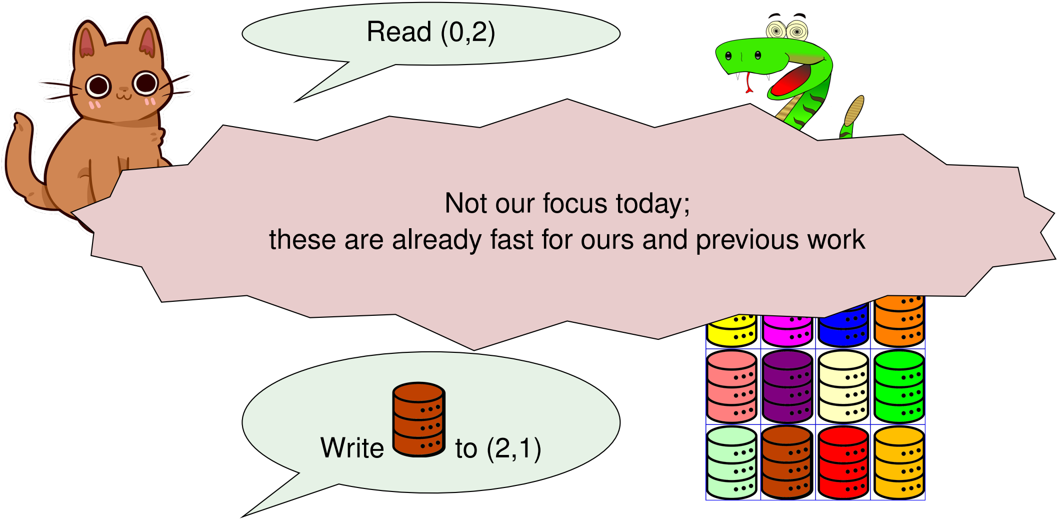
Basic Operations: Read and Update (hence *Dynamic*)



Basic Operations: Read and Update (hence *Dynamic*)



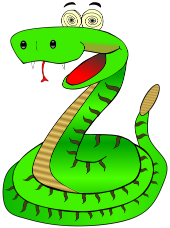
Basic Operations: Read and Update (hence *Dynamic*)

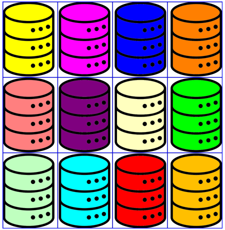


Level-0 Audit: Nothing

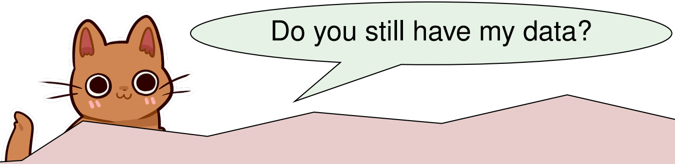
Do you still have my data?

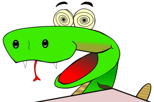
Yes, trust me!





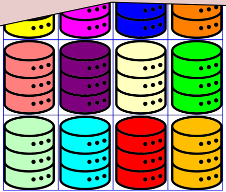
Level-0 Audit: Nothing



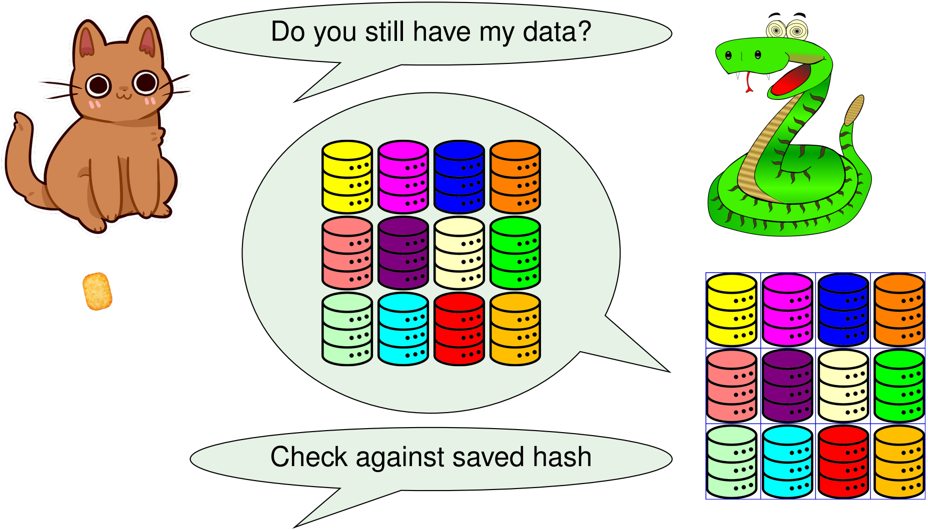


Current practice for AWS, MS Azure, etc. : Security is only by Reputation

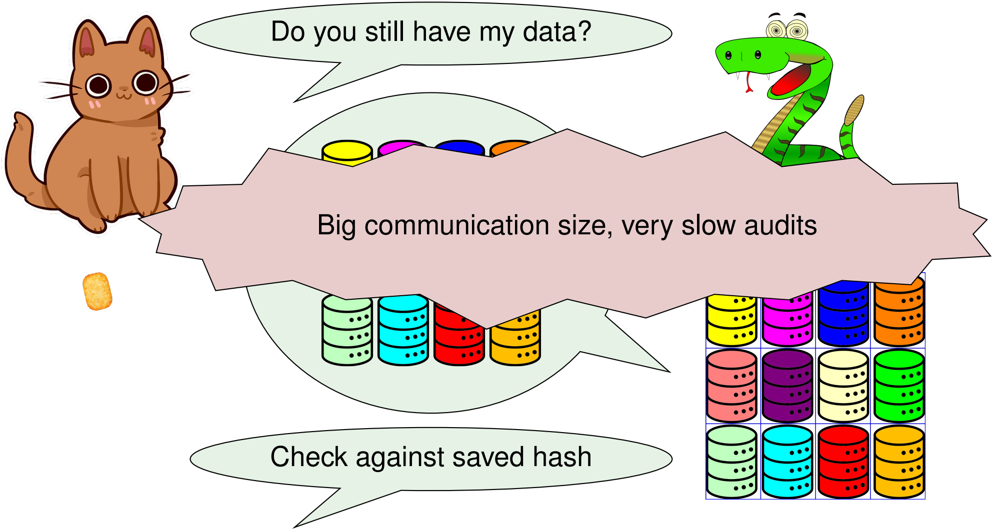
⚠ Problem for Decentralized Storage Networks such as **FileCoin** . . .



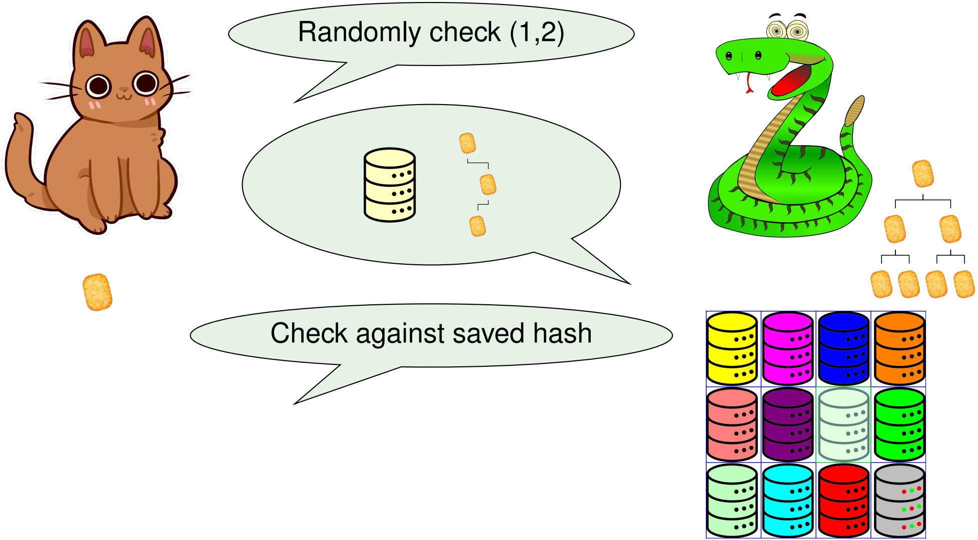
Level-1 Audit: Trivial



Level-1 Audit: Trivial

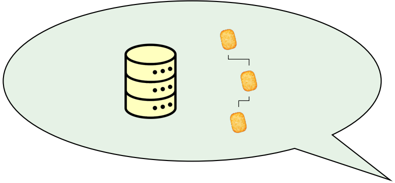


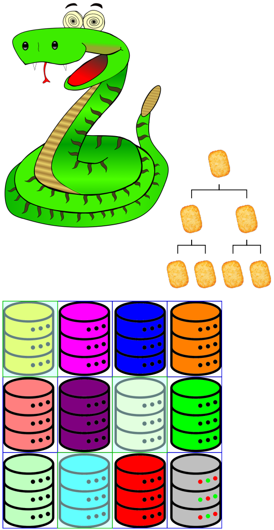
Level-2 Audit: Provable Data Possession (PDP)



Level-2 Audit: Provable Data Possession (PDP)

Randomly check (1,2)

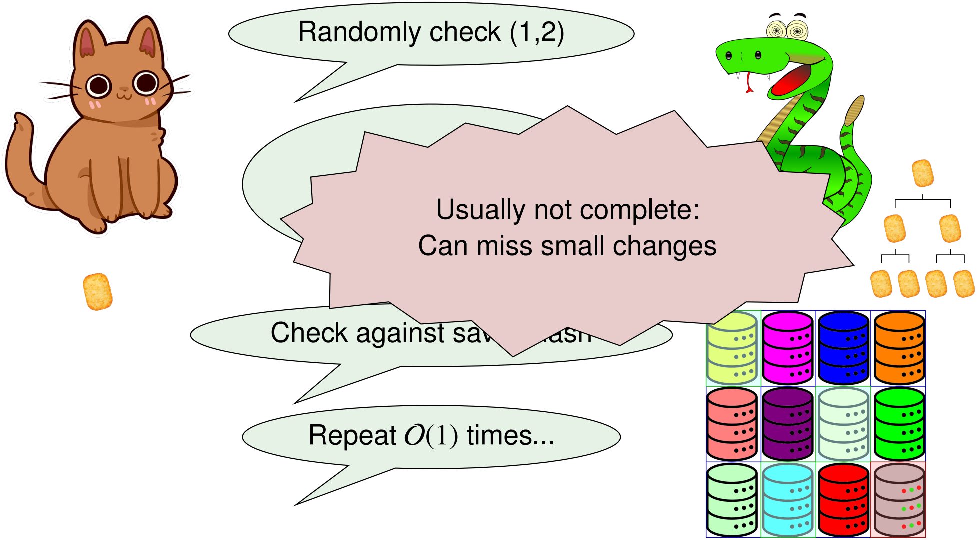




Check against saved hash

Repeat $\mathcal{O}(1)$ times...

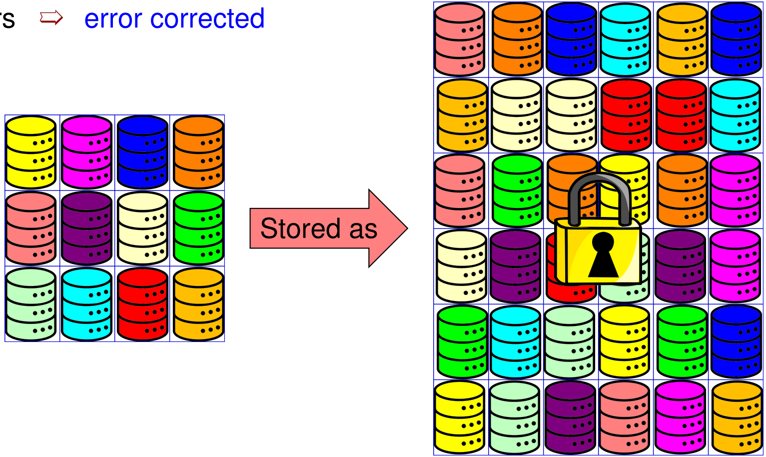
Level-2 Audit: Provable Data Possession (PDP)



Proof of Retrievability (PoR) Storage

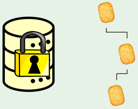
Idea (*[Cash et al '13]*, *[Shi et al '13]*): Redundancy, shuffling, and encryption

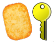
- Large errors ⇨ caught by random checks
- Small errors ⇨ error corrected



Level-3 Audit: Proof of Retrievability (PoR)

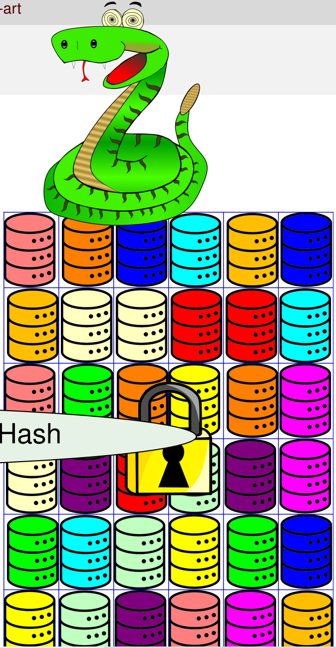
Randomly check (3,0)





Decrypt, Decode and check against saved Hash

Repeat $O(1)$ times...

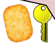


Level-3 Audit: Proof of Retrievability (PoR)

Randomly check (3,0)

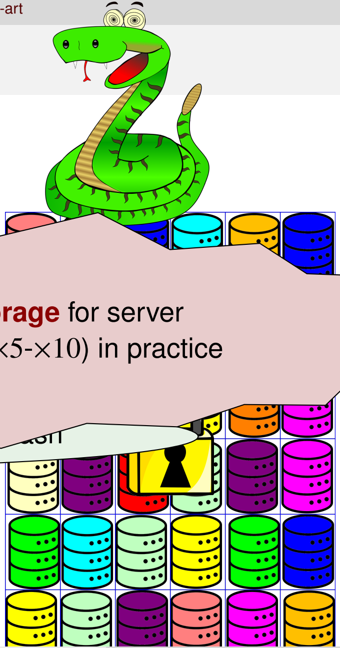
Non-transparent and **large storage** for server

High **permanent** monetary cost ($\times 5$-$\times 10$) in practice



Decrypt, Decode and c... inst sav... ash

Repeat $O(1)$ times...



Existing Work Comparison Summary

	Trivial	DPDP	DPoR
Fast audit (client)	✗	✓	✓
Fast audit (server)	✗	✓	✓
Complete audit	✓	✗	✓
Transparent storage	✓	✓	✗

Existing Work Comparison Summary

	Trivial	DPDP	DPoR
Fast audit (client)	✗	✓	✓
Fast audit (server)	✗	✓	✓
Complete audit	✓	✗	✓
Transparent storage	✓	✓	✗

1. You can't have it all: $$\text{(extra storage size)} \cdot \frac{\text{audit cost}}{\log(\text{audit cost})} \in \Omega(\text{data size})$$

   [ADHJMPR, Dynamic Proofs of Retrievability with Low Server Storage (Usenix SECURITY 2021)]

Existing Work Comparison Summary

	Trivial	DPDP	DPoR	[ADHJMPR]
Fast audit (client)	✗	✓	✓	✓
Fast audit (server)	✗	✓	✓	✗
Complete audit	✓	✗	✓	✓
Transparent storage	✓	✓	✗	✓

1. You can't have it all: $(\text{extra storage size}) \cdot \dfrac{\text{audit cost}}{\log(\text{audit cost})} \in \Omega(\text{data size})$
2. New constructions with different trade-off
3. Practical deployment on a commercial cloud
   - Computations are usually much cheaper than long-term storage!

Outline

New Strategy for Audits

- Treat data as a $O\left(\sqrt{N}\right) \times O\left(\sqrt{N}\right)$ matrix, in-place
- Client computes a random linear combination of rows during initialization
- For audits:
  1. Client chooses a random control vector
  2. Server computes corresponding random linear combination of columns
  3. Client checks two dot products for equality

New Strategy for Audits

- Treat data as a $O\left(\sqrt{N}\right) \times O\left(\sqrt{N}\right)$ matrix, in-place
- Client computes a random linear combination of rows during initialization
- For audits:
  1. Client chooses a random control vector
  2. Server computes corresponding random linear combination of columns
  3. Client checks two dot products for equality

Lemma (R. Freivalds, "Probabilistic Machines Can Use Less Running Time", 1977)

For any matrices $\mathbf{A}$, $\mathbf{B}$ *and random vector* $\mathbf{x}$ *over a large enough field,*
$\mathbf{A} \neq \mathbf{B}$ *implies* $\mathbf{Ax} \neq \mathbf{Bx}$ *with high probability.*

New Strategy for Audits

- Treat data as a $O\left(\sqrt{N}\right) \times O\left(\sqrt{N}\right)$ matrix, in-place
- Client computes a random linear combination of rows during initialization
- For audits:
  1. Client chooses a random control vector
  2. Server computes corresponding random linear combination of columns
  3. Client checks two dot products for equality

Lemma (R. Freivalds, "Probabilistic Machines Can Use Less Running Time", 1977)

For any matrices $\mathbf{A}$, $\mathbf{B}$ *and random vectors* $\mathbf{u}$, $\mathbf{x}$ *over a large enough field,*
$\mathbf{A} \neq \mathbf{B}$ ***implies*** $(\mathbf{u}^\intercal \mathbf{A})\mathbf{x} \neq \mathbf{u}^\intercal(\mathbf{B}\mathbf{x})$ *with high probability.*

Protocol 1: Privately-verifiable computations for Audits

	Client  Communications  Server
Init	Secret $\mathbf{u}$

Protocol 1: Privately-verifiable computations for Audits

	Client  Communications  Server
Init	Secret $\mathbf{u}$ Secret $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$

Protocol 1: Privately-verifiable computations for Audits

	Client	Communications	Server
Init	Secret $\mathbf{u}$ Secret $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$	$\xrightarrow{\mathbf{A}}$	

Protocol 1: Privately-verifiable computations for Audits

	Client	Communications	Server
Init	Secret $\mathbf{u}$ Secret $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$	$\xrightarrow{\mathbf{A}}$	
Audit	Random $\mathbf{x}$	$\xrightarrow{\mathbf{x}}$	

Protocol 1: Privately-verifiable computations for Audits

	Client	Communications	Server
Init	Secret $\mathbf{u}$		
	Secret $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$	$\xrightarrow{\mathbf{A}}$	
Audit	Random $\mathbf{x}$	$\xrightarrow{\mathbf{x}}$	$\mathbf{y} = \mathbf{A}\mathbf{x}$
		$\xleftarrow{\mathbf{y}}$	

Protocol 1: Privately-verifiable computations for Audits

	Client 	Communications	 Server
Init	Secret $\mathbf{u}$		
	Secret $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$	$\xrightarrow{\mathbf{A}}$	
Audit	Random $\mathbf{x}$	$\xrightarrow{\mathbf{x}}$	$\mathbf{y} = \mathbf{A}\mathbf{x}$
	checks $\mathbf{v}^\intercal \mathbf{x} \stackrel{?}{=} \mathbf{u}^\intercal \mathbf{y}$	$\xleftarrow{\mathbf{y}}$	

Protocol 1: Privately-verifiable computations for Audits

	Client	Communications	Server
Init	Secret $\mathbf{u}$		
	Secret $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$	$\xrightarrow{\mathbf{A}}$	
Audit	Random $\mathbf{x}$	$\xrightarrow{\mathbf{x}}$	$\mathbf{y} = \mathbf{A}\mathbf{x}$
	checks $\mathbf{v}^\intercal \mathbf{x} \stackrel{?}{=} \mathbf{u}^\intercal \mathbf{y}$	$\xleftarrow{\mathbf{y}}$	

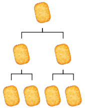
$$\begin{cases} \mathbf{v}^\intercal \mathbf{x} = \mathbf{u}^\intercal \mathbf{A}\mathbf{x} = \mathbf{u}^\intercal \mathbf{y} & \Leftarrow \quad \text{unmodified } \mathbf{A} \\ \mathbf{v}^\intercal \mathbf{x} = \mathbf{u}^\intercal \mathbf{A}\mathbf{x} \neq \mathbf{u}^\intercal \mathbf{y}' & \Leftarrow \quad \text{w.h.p., otherwise} \end{cases}$$

Protocol 1: Privately-verifiable computations for Audits

	Client	Communications	Server
Init	Secret $\mathbf{u}$		
	Secret $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$	$\xrightarrow{\mathbf{A}}$	
Audit	Random $\mathbf{x}$	$\xrightarrow{\mathbf{x}}$	$\mathbf{y} = \mathbf{A}\mathbf{x}$
	checks $\mathbf{v}^\intercal \mathbf{x} \overset{?}{=} \mathbf{u}^\intercal \mathbf{y}$	$\xleftarrow{\mathbf{y}}$	

$$\begin{cases} \mathbf{v}^\intercal \mathbf{x} = \mathbf{u}^\intercal \mathbf{A}\mathbf{x} = \mathbf{u}^\intercal \mathbf{y} & \Leftarrow \quad \text{unmodified } \mathbf{A} \\ \mathbf{v}^\intercal \mathbf{x} = \mathbf{u}^\intercal \mathbf{A}\mathbf{x} \neq \mathbf{u}^\intercal \mathbf{y}' & \Leftarrow \quad \text{w.h.p., otherwise} \end{cases}$$

✚ **Merkle Hash trees**



For efficient & verified:

- **Read/Write** of $\mathbf{A}$
- **Update** of $\mathbf{v}$: $\mathbf{v}'_j \leftarrow \mathbf{v}_j + \mathbf{u}_i(\mathbf{A}'_{ij} - \mathbf{A}_{ij})$

Formal security

Statistical security, even in the presence of a **malicious** server:

Theorem (Security)

- ***Correct***: *With an honest client and an honest server, audits are accepted & reads recover the last updated values of the database;*
- ***Verifiable***: *The client can always detect, except with negligible probability, if any message even sent by a malicious server deviates from honest behavior ;*
- ***Retreivable***: *In order to pass an audit test with high probability, a malicious server has to have access to the entire memory contents.*

Formal security

Statistical security, even in the presence of a **malicious** server:

Theorem (Security)

- ***Correct***: *With an honest client and an honest server, audits are accepted & reads recover the last updated values of the database;*
- ***Verifiable***: *The client can always detect, except with negligible probability, if any message even sent by a malicious server deviates from honest behavior ;*
- ***Retreivable***: *In order to pass an audit test with high probability, a malicious server has to have access to the entire memory contents.*

⇨ For $2^{-\lambda}$ probability of failure: consider DB as a $\sqrt{N/\lambda} \times \sqrt{N/\lambda}$ matrix over λ-bits prime field

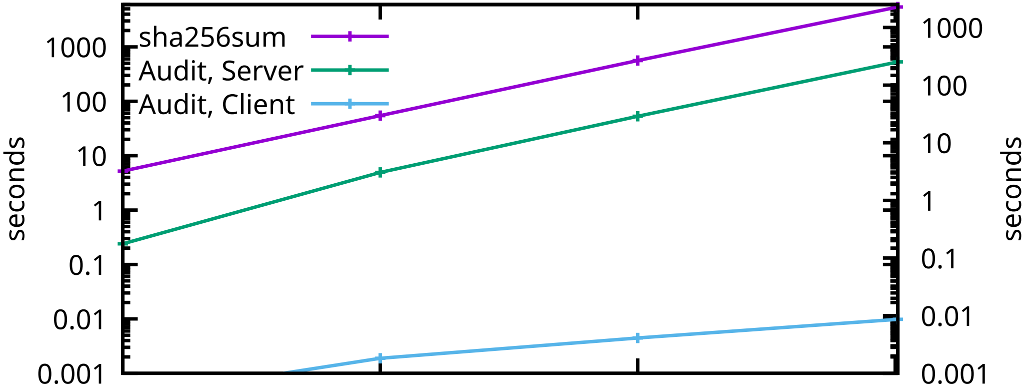
⇨ $O\left(\sqrt{\lambda N}\right)$ client secret **storage**, audit **communication** & **computations**

Experimental Design

- Open-source implementation written in C using OpenSSL and OpenMP
- Tested on Google Cloud Compute
  - Client: `f1-micro` shared CPU VM in Belgium
  - Server: `n1-standard-2` single-CPU VM in Iowa, with attached Local SSD storage
- Data: random files of size **1GB, 10GB, 100GB, 1TB**
- Testing performed in May 2021

Open-source client-server code: https://github.com/dsroche/la-por

Google Cloud Compute (Belgium ⇋ Iowa)



Database:	1GB	10GB	100GB	1TB
Proof size:	0.2MB	0.5MB	1.7MB	5.4MB
Client keys:	0.2MB	0.5MB	1.7MB	5.4MB

Outline

Further improvements?

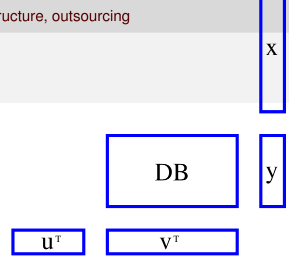
Client Storage (keys): $\mathbf{u}$ and $\mathbf{v}$

Communications (proof size): $\mathbf{x}$ and $\mathbf{y}$

Client time (computations): $\boxed{\mathbf{v}^\intercal \mathbf{x} \stackrel{?}{=} \mathbf{u}^\intercal \mathbf{y}}$

$\mathcal{O}\left(\sqrt{N}\right)$ might still be too much, e.g., for Decentralized Storage Networks . . .

Further improvements?



Client Storage (keys): $\mathbf{u}$ and $\mathbf{v}$

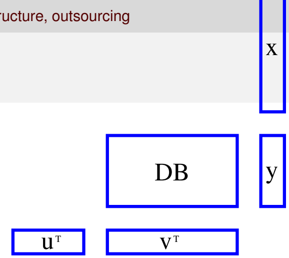
Communications (proof size): $\mathbf{x}$ and $\mathbf{y}$

Client time (computations): $\mathbf{v}^\intercal \mathbf{x} \stackrel{?}{=} \mathbf{u}^\intercal \mathbf{y}$

$\mathcal{O}(\sqrt{N})$ might still be too much, e.g., for Decentralized Storage Networks . . .

1. Rectangular database: small, $\mathcal{O}(\log(N))$, $\mathbf{u}$ and $\mathbf{y}$

Further improvements?



Client Storage (keys): $\mathbf{u}$ and $\mathbf{v}$

Communications (proof size): $\mathbf{x}$ and $\mathbf{y}$

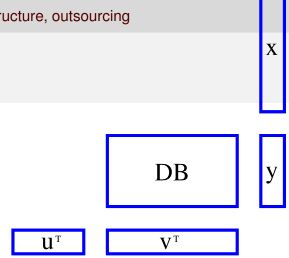
Client time (computations): $\mathbf{v}^\intercal \mathbf{x} \stackrel{?}{=} \mathbf{u}^\intercal \mathbf{y}$

$\mathcal{O}(\sqrt{N})$ might still be too much, e.g., for Decentralized Storage Networks . . .

1. Rectangular database: small, $\mathcal{O}(\log(N))$, $\mathbf{u}$ and $\mathbf{y}$
2. Structure: $\mathbf{u} = [1, \mu, \mu^2, \ldots, \mu^{m-1}]$ and $\mathbf{x} = [1, r, r^2, \ldots, r^{n-1}]$, $\mathcal{O}(1)$

   ⇨ from **dotproducts** to polynomial evaluation

Further improvements?



Client Storage (keys): $\mathbf{u}$ and $\mathbf{v}$

Communications (proof size): $\mathbf{x}$ and $\mathbf{y}$

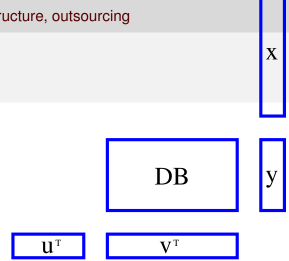
Client time (computations): $\mathbf{v}^\intercal\mathbf{x} \stackrel{?}{=} \mathbf{u}^\intercal\mathbf{y}$

$O(\sqrt{N})$ might still be too much, e.g., for Decentralized Storage Networks ...

1. Rectangular database: small, $O(\log(N))$, $\mathbf{u}$ and $\mathbf{y}$
2. Structure: $\mathbf{u} = [1, \mu, \mu^2, \ldots, \mu^{m-1}]$ and $\mathbf{x} = [1, r, r^2, \ldots, r^{n-1}]$, $O(1)$

   ⇨ from **dotproducts** to polynomial evaluation
3. Store $\mathbf{v}$, encrypted as $\mathbf{w} = E(\mathbf{v})$, on Server

Further improvements?



Client Storage (keys): $\mathbf{u}$ and $\mathbf{v}$

Communications (proof size): $\mathbf{x}$ and $\mathbf{y}$

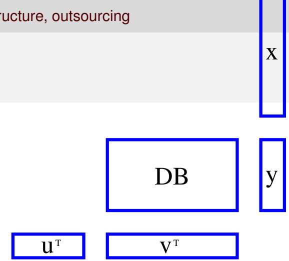
Client time (computations): $\mathbf{v}^\intercal\mathbf{x} \stackrel{?}{=} \mathbf{u}^\intercal\mathbf{y}$

$\mathcal{O}(1)$, $\mathcal{O}(\log N)$, $\mathcal{O}(\log N)$

1. Rectangular database: small, $\mathcal{O}(\log(N))$, $\mathbf{u}$ and $\mathbf{y}$
2. Structure: $\mathbf{u} = [1, \mu, \mu^2, \ldots, \mu^{m-1}]$ and $\mathbf{x} = [1, r, r^2, \ldots, r^{n-1}]$, $\mathcal{O}(1)$
   ⇨ from **dotproducts** to polynomial evaluation
3. Store $\mathbf{v}$, encrypted as $\mathbf{w} = E(\mathbf{v})$, on Server
4. **Outsource & Verify**, homomorphic $\mathbf{w}^\intercal \odot \mathbf{x} = E(P_\mathbf{v}(r))$, on Server

$\{P_v(r) = \sum v_i r^i\}$

Further improvements?



Client Storage (keys): $\mathbf{u}$ and $\mathbf{v}$

Communications (proof size): $\mathbf{x}$ and $\mathbf{y}$

Client time (computations): $\mathbf{v}^\intercal \mathbf{x} \stackrel{?}{=} \mathbf{u}^\intercal \mathbf{y}$

$\mathcal{O}(1)$, $\mathcal{O}(\log N)$, $\mathcal{O}(\log N)$

1. Rectangular database: small, $\mathcal{O}(\log(N))$, $\mathbf{u}$ and $\mathbf{y}$
2. Structure: $\mathbf{u} = [1, \mu, \mu^2, \ldots, \mu^{m-1}]$ and $\mathbf{x} = [1, r, r^2, \ldots, r^{n-1}]$, $\mathcal{O}(1)$
   ⇨ from **dotproducts** to polynomial evaluation
3. Store $\mathbf{v}$, encrypted as $\mathbf{w} = E(\mathbf{v})$, on Server
4. **Outsource & Verify**, homomorphic $\mathbf{w}^\intercal \odot \mathbf{x} = E(P_\mathbf{v}(r))$, on Server $\{P_v(r) = \sum v_i r^i\}$

⇨ *[DMPR, VESPo: Verified Evaluation of Secret Polynomials (PoPETS 2023)]*

Verified evaluation of secret dynamic polynomials

Issues:

1. **Security**: Soundness (evaluation binding) + Privacy (hiding)
2. **Dynamicity**: fast partial updates + without new weaknesses
3. **Efficiency**: fast Client + practical Server

Verified evaluation of secret dynamic polynomials

Issues:

1. **Security**: Soundness (evaluation binding) + Privacy (hiding)
2. **Dynamicity**: fast partial updates + without new weaknesses
3. **Efficiency**: fast Client + practical Server

Privacy: hiding via efficient $2D$-geom. masking of P, in the exponents

- $(2, 1, d)$-**DLM** security assumption ($\approx$*Decision Linear*, if $\exists$ pairing)
  [Abdalla et al. Crypto 2015]

Verified evaluation of secret dynamic polynomials

Issues:

1. **Security**: Soundness (evaluation binding) + Privacy (hiding)
2. **Dynamicity**: fast partial updates + without new weaknesses
3. **Efficiency**: fast Client + practical Server

Privacy: hiding via efficient $2D$-geom. masking of P, in the exponents

- $(2, 1, d)$-**DLM** security assumption (≈*Decision Linear*, if ∃ pairing)
  [Abdalla et al. Crypto 2015]

$\alpha \xleftarrow{\$} \mathbb{Z}_p^2, \quad \beta \xleftarrow{\$} \mathbb{Z}_p^2, \quad \mathbf{\Phi} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$ geom. masking $\boxed{\bar{P}(X) \leftarrow P(X)\alpha + \Gamma(X)\beta} = \sum_{i=0}^{d} X^i(p_i\alpha + \mathbf{\Phi}^i\beta)$

Verified evaluation of secret dynamic polynomials

Issues:

1. **Security**: Soundness (evaluation binding) + Privacy (hiding)
2. **Dynamicity**: fast partial updates + without new weaknesses
3. **Efficiency**: fast Client + practical Server

Privacy: hiding via efficient $2D$-geom. masking of P, in the exponents

- $(2, 1, d)$-**DLM** security assumption (≈*Decision Linear*, if ∃ pairing)
  [Abdalla et al. Crypto 2015]

$\alpha \xleftarrow{\$} \mathbb{Z}_p^2, \quad \beta \xleftarrow{\$} \mathbb{Z}_p^2, \quad \mathbf{\Phi} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$ geom. masking $\boxed{\bar{P}(X) \leftarrow P(X)\alpha + \Gamma(X)\beta} = \sum_{i=0}^{d} X^i(p_i\alpha + \mathbf{\Phi}^i\beta)$

- Client Efficiency ⇨ unmasking via $\boxed{\Gamma(r)\beta = \left(\frac{(r\mathbf{\Phi})^{d+1} - I_2}{r\mathbf{\Phi} - I_2}\right)\beta} = \sum_{i=0}^{d} r^i \mathbf{\Phi}^i \beta$ *[Fiduccia]*

Verified evaluation of secret dynamic polynomials

Issues:

1. **Security**: Soundness (evaluation binding) + Privacy (hiding)
2. **Dynamicity**: fast partial updates + without new weaknesses
3. **Efficiency**: fast Client + practical Server

Soundness: Evaluation binding

- Difference polynomial, check $P(r)$ with precomputed secret evaluation $P(s)$:

$$\boxed{P(s) = P(r) + (s-r)\left(\frac{P(X)-P(Y)}{X-Y}\right)(s,r) = P(r) + (s-r)Q_P(s,r)} \tag{1}$$

Verified evaluation of secret dynamic polynomials

Issues:

1. **Security**: Soundness (evaluation binding) + Privacy (hiding)
2. **Dynamicity**: fast partial updates + without new weaknesses
3. **Efficiency**: fast Client + practical Server

Soundness: Evaluation binding

- Difference polynomial, check $P(r)$ with precomputed secret evaluation $P(s)$:

$$\boxed{P(s) = P(r) + (s-r)\left(\frac{P(X)-P(Y)}{X-Y}\right)(s,r) = P(r) + (s-r)Q_P(s,r)} \tag{1}$$

- Bilinear Pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, generated by $g_1, g_2, g_T = e(g_1; g_2)$

⇨ Server Homorphically computes $g_T^{Q_P(s,r)} \ldots$ 👍 linear (✚ linear precomputations)

Verified evaluation of secret dynamic polynomials

Issues:

1. **Security**: Soundness (evaluation binding) + Privacy (hiding)
2. **Dynamicity**: fast partial updates + without new weaknesses
3. **Efficiency**: fast Client + practical Server

Soundness: Evaluation binding

- Difference polynomial, check $P(r)$ with precomputed secret evaluation $P(s)$:

$$P(s) = P(r) + (s - r)\left(\frac{P(X) - P(Y)}{X - Y}\right)(s, r) = P(r) + (s - r)Q_P(s, r) \tag{1}$$

- Bilinear Pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, generated by $g_1, g_2, g_T = e(g_1; g_2)$

⇨ Server Homorphically computes $g_T^{Q_P(s,r)} \ldots$ 👍 linear (✚ linear precomputations)

⇨ Client Homorphically checks Equation (1) in $\mathbb{G}_T$

Verification in the exponents

Goal ⇨ have the server compute: $\boxed{\zeta = E(P(r))}$, via linear homomorphic encryption (LHE)

Verification in the exponents

Goal ⇨ have the server compute: $\boxed{\zeta = E(P(r))}$, via linear homomorphic encryption (LHE)

Verify $\boxed{\zeta}$, using, in the exponents: $\boxed{P(s) = P(r) + Q_P(s,r)(s-r)}$, via pairings

Verification in the exponents

Goal ⇨ have the server compute: $\boxed{\zeta = E(P(r))}$, via linear homomorphic encryption (LHE)

Verify $\boxed{\zeta}$, using, in the exponents: $\boxed{P(s) = P(r) + Q_P(s,r)(s-r)}$, via pairings

	Client 	Communications	 Server
Init	$\mathbf{w} \leftarrow E(P)$, ciphered $\mathcal{K} \leftarrow g_T^{P(s)}$	$\xrightarrow{\mathbf{w}}$	

Verification in the exponents

Goal ⇨ have the server compute: $\boxed{\zeta = E(P(r))}$, via linear homomorphic encryption (LHE)

Verify $\boxed{\zeta}$, using, in the exponents: $\boxed{P(s) = P(r) + Q_P(s,r)(s-r)}$, via pairings

	Client	Communications	Server
Init	$\mathbf{w} \leftarrow E(P)$, ciphered $\mathcal{K} \leftarrow g_T^{P(s)}$	$\xrightarrow{\mathbf{w}}$	
Audit	Random point r	$\xrightarrow{r}$	

Verification in the exponents

Goal ⇨ have the server compute: $\boxed{\zeta = E(P(r))}$, via linear homomorphic encryption (LHE)

Verify $\boxed{\zeta}$, using, in the exponents: $\boxed{P(s) = P(r) + Q_P(s,r)(s-r)}$, via pairings

	Client	Communications	Server
Init	$\mathbf{w} \leftarrow E(P)$, ciphered $\mathcal{K} \leftarrow g_T^{P(s)}$	$\xrightarrow{\mathbf{w}}$	
Audit	Random point r	$\xrightarrow{r}$	$\zeta = E(P) \odot [r^i]$

{homomorphic}

Verification in the exponents

Goal ⇨ have the server compute: $\boxed{\zeta = E(P(r))}$, via linear homomorphic encryption (LHE)

Verify $\boxed{\zeta}$, using, in the exponents: $\boxed{P(s) = P(r) + Q_P(s,r)(s-r)}$, via pairings

	Client 	Communications	 Server	
Init	$\mathbf{w} \leftarrow E(P)$, ciphered	$\xrightarrow{\mathbf{w}}$		
	$\mathcal{K} \leftarrow g_T^{P(s)}$			
Audit	Random point r	$\xrightarrow{r}$	$\zeta = E(P) \odot [r^i]$	{homomorphic}
		$\xleftarrow{\zeta, \xi}$	$\xi = g_T^{Q_P(s,r)}$	{certificate}

Verification in the exponents

Goal ⇨ have the server compute: $\boxed{\zeta = E(P(r))}$, via linear homomorphic encryption (LHE)

Verify $\boxed{\zeta}$, using, in the exponents: $\boxed{P(s) = P(r) + Q_P(s,r)(s-r)}$, via pairings

	Client	Communications	Server	
Init	$\mathbf{w} \leftarrow E(P)$, ciphered	$\xrightarrow{\mathbf{w}}$		
	$\mathcal{K} \leftarrow g_T^{P(s)}$			
Audit	Random point r	$\xrightarrow{r}$	$\zeta = E(P) \odot [r^i]$	{homomorphic}
	checks $\mathcal{K} \stackrel{?}{=} g_T^{D(\zeta)} \xi^{s-r}$	$\xleftarrow{\zeta, \xi}$	$\xi = g_T^{Q_P(s,r)}$	{certificate}

☞ $\mathcal{K} = g_T^{P(s)}$ should be $g_T^{D(\zeta)} \xi^{s-r} = g_T^{P(r)+Q_P(s,r)(s-r)}$

Verification in the exponents

Goal ⇨ have the server compute: $\boxed{\zeta = E(P(r))}$, via linear homomorphic encryption (LHE)

Verify $\boxed{\zeta}$, using, in the exponents: $\boxed{P(s) = P(r) + Q_P(s,r)(s-r)}$, via pairings

	Client	Communications	Server	
Init	$\mathbf{w} \leftarrow E(P)$, ciphered	$\xrightarrow{\mathbf{w}}$		
	$\mathcal{K} \leftarrow g_T^{P(s)}$			
Audit	Random point r	$\xrightarrow{r}$	$\zeta = E(P) \odot [r^i]$	{homomorphic}
	checks $\mathcal{K} \stackrel{?}{=} g_T^{D(\zeta)} \xi^{s-r}$	$\xleftarrow{\zeta, \xi}$	$\xi = g_T^{Q_P(s,r)}$	{certificate}

☞ $\mathcal{K} = g_T^{P(s)}$ should be $g_T^{D(\zeta)} \xi^{s-r} = g_T^{P(r)+Q_P(s,r)(s-r)}$

⚠ How can the Server efficiently & securely compute $\boxed{\xi = g_T^{Q_P(s,r)}}$?

Fast Horner-like certification using a pairing

Server: has to compute $\boxed{\xi = g_T^{Q_P(s,r)}}$

Fast Horner-like certification using a pairing

Server: has to compute $\boxed{\xi = g_T^{Q_P(s,r)}}$

Lemma

If $P(X) = \sum_{i=0}^{d} p_i X^i$, *then*

$$Q_P(s,r) = \sum_{i=1}^{d} \sum_{k=0}^{i-1} p_i s^{i-k-1} r^k$$

Fast Horner-like certification using a pairing

Server: has to compute $\boxed{\xi = g_T^{Q_P(s,r)}}$

Lemma

If $P(X) = \sum_{i=0}^{d} p_i X^i$*, then*

$$Q_P(s, r) = \sum_{i=1}^{d} \sum_{k=0}^{i-1} p_i s^{i-k-1} r^k$$

Sum of 3-terms products:

⚠ quadratic?

Fast Horner-like certification using a pairing

Server: has to compute $\xi = g_T^{Q_P(s,r)}$

Lemma

If $P(X) = \sum_{i=0}^{d} p_i X^i$, *then*

$$Q_P(s,r) = \sum_{i=1}^{d} \sum_{k=0}^{i-1} p_i s^{i-k-1} r^k$$

$$\begin{array}{lrll} i=1 & (s^0 r^0) & \cdot p_1 & + \\ i=2 & (s^1 r^0 + s^0 r^1) & \cdot p_2 & + \\ i=3 & (s^2 r^0 + s^1 r^1 + s^0 r^2) & \cdot p_3 & + \\ \ldots & & & \end{array}$$

Sum of 3-terms products:

⚠ quadratic?

Fast Horner-like certification using a pairing

Server: has to compute $\xi = g_T^{Q_P(s,r)}$

Lemma

If $P(X) = \sum_{i=0}^{d} p_i X^i$, *then*

$$Q_P(s,r) = \sum_{i=1}^{d} \sum_{k=0}^{i-1} p_i s^{i-k-1} r^k$$

$$\begin{array}{lrcl} i = 1 & (s^0 r^0) & \cdot p_1 & + \\ i = 2 & (s^1 r^0 + (s^0 r^0) r) & \cdot p_2 & + \\ i = 3 & (s^2 r^0 + (s^1 r^0 + s^0 r^1) r) & \cdot p_3 & + \\ \dots & & & \end{array}$$

Sum of 3-terms products:

⚠ quadratic?

Fast Horner-like certification using a pairing

Server: has to compute $\boxed{\xi = g_T^{Q_P(s,r)}}$

Lemma

If $P(X) = \sum_{i=0}^{d} p_i X^i$, *then*

$$Q_P(s,r) = \sum_{i=1}^{d} \sum_{k=0}^{i-1} p_i s^{i-k-1} r^k$$

Sum of 3-terms products:

⚠ quadratic? ⇨ linear!

$$\begin{array}{lrll} i=1 & (s^0 r^0) & \cdot p_1 & + \\ i=2 & (s^1 r^0 + (s^0 r^0) r) & \cdot p_2 & + \\ i=3 & (s^2 r^0 + (s^1 r^0 + s^0 r^1) r) & \cdot p_3 & + \\ \dots & & & \end{array}$$

Algorithm Compute $Q_P(s,r)$ in **clear**

$t \leftarrow 0, z \leftarrow 0$
for $i = 1 \dots d$ **do**
  $t \leftarrow s^{i-1} + t \times r$
  $z \leftarrow z + t \times p_i$
end for
return z

Fast Horner-like certification using a pairing

Server: has to compute $\boxed{\xi = g_T^{Q_P(s,r)}}$

Lemma

If $P(X) = \sum_{i=0}^{d} p_i X^i$, *then*

$$Q_P(s,r) = \sum_{i=1}^{d} \sum_{k=0}^{i-1} p_i s^{i-k-1} r^k$$

Sum of 3-terms products:

- ⚠ quadratic? ⇨ linear!
- ⚠ not linearly homomorphic?
- ⇨ $(p_* s^*) \times r^*$ using **ciphered×clear** product
- ⇨ $p_* \times s^*$ using a **pairing**

Algorithm Compute $Q_P(s,r)$ in exponents

$t \leftarrow 1_{\mathbb{G}_2}, \xi \leftarrow 1_{\mathbb{G}_T}$

for $i = 1 \ldots d$ **do**

  $t \leftarrow g_1^{s^{i-1}} \cdot t^r$

  $\xi \leftarrow \xi \cdot \mathbf{e}(t; g_2^{p_i})$

end for

return ξ

Fast Horner-like certification using a pairing

Server: has to compute $\boxed{\xi = g_T^{Q_P(s,r)}}$

Lemma

If $P(X) = \sum_{i=0}^{d} p_i X^i$, *then*

$$Q_P(s,r) = \sum_{i=1}^{d} \sum_{k=0}^{i-1} p_i s^{i-k-1} r^k$$

Sum of 3-terms products:

⚠ quadratic? ⇨ linear!

⚠ not linearly homomorphic?

⇨ $(p_* s^*) \times r^*$ using **ciphered×clear** product

⇨ $p_* \times s^*$ using a **pairing**

Init Client 

$S \leftarrow [g_1^{s^k}]_{k=0..d-1}$

$H \leftarrow [g_2^{p_i}]_{i=1..d}$

S, H ⇢

Algorithm Compute $Q_P(s,r)$ in ciphertext

$t \leftarrow 1_{\mathbb{G}_2}, \xi \leftarrow 1_{\mathbb{G}_T}$

for $i = 1 \ldots d$ **do**

  $t \leftarrow \boxed{S_{i-1}} \cdot t^r$

  $\xi \leftarrow \xi \cdot \mathbf{e}(\mathbf{t}; \boxed{H_i})$

end for

return ξ

Processor oblivious Parallel Server

$$\text{degree } d \approx (b \text{ blocks}) \times (q \text{ elements})$$

- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$

Processor oblivious Parallel Server

degree $d \approx (b$ blocks$) \times (q$ elements$)$

- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$

  1. Parallel geometric progression

$$\boxed{[\rho_i] = [\ldots, \langle r^5, \ldots, r^8\rangle, \langle r^9, \ldots, \ldots, r^{16}\rangle, \ldots]}$$ $\{\log_2(d)$ parallel steps$\}$

Processor oblivious Parallel Server

$$\text{degree } d \approx (b \text{ blocks}) \times (q \text{ elements})$$

- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations (generalized Strauß-Shamir trick)

- **parfor** $k = 1..q$ **do** $\boxed{\zeta_k \leftarrow \prod_{i=b_{k-1}}^{b_k-1} w_i^{\rho_i}}$ **endparfor** $\{q$ blocks in parallel$\}$
- Parallel associative reduction: $\boxed{\zeta \leftarrow \prod_{k=1}^{q} \zeta_k}$ $\{\log_2(q)$ parallel steps$\}$

Processor oblivious Parallel Server

degree $d \approx (b \text{ blocks}) \times (q \text{ elements})$

- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations (generalized Strauß-Shamir trick)
- Certificate : $\xi = g_T^{Q_P(s,r)} = \prod_{i=1}^{d} \prod_{k=0}^{i-1} e(S_{i-k-1}; \bar{H}_i[j])^{\rho_k}$

Processor oblivious Parallel Server

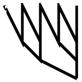
degree $d \approx (b$ blocks$) \times (q$ elements$)$

- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations (generalized Strauß-Shamir trick)
- Certificate : $\xi = g_T^{Q_P(s,r)} = \prod_{i=1}^{d} \prod_{k=0}^{i-1} e(S_{i-k-1}; \bar{H}_i[j])^{\rho_k}$
  3. Parallel prefix-like, Horner-like on all $S_{i-k-1}^{\rho_k}$
     - $\boxed{u_\ell = \prod_{k=0}^{\ell} S_{\ell-k}^{\rho_k}}$, for $\ell = 0..(d-1)$ ⇨ Family of binary gates $\theta_{\rho_i}(a,b) = a \cdot b^{\rho_i}$
     - Optimal lower bound: **Work** $\geq d\left(2 - \frac{1}{p}\right)$ on p processors [Snir'86]

Processor oblivious Parallel Server

degree $d \approx (b \text{ blocks}) \times (q \text{ elements})$

- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations

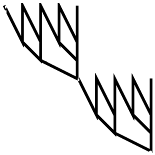


- Certificate : $\xi = g_T^{Q_P(s,r)} = \prod_{i=1}^{d} \prod_{k=0}^{i-1} e(S_{i-k-1}; \bar{H}_i[j])^{\rho_k}$
  3. Parallel prefix-like, Horner-like on all $S_{i-k-1}^{\rho_k}$

Processor oblivious Parallel Server

degree $d \approx (b \text{ blocks}) \times (q \text{ elements})$

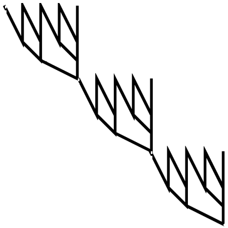
- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations
- Certificate : $\xi = g_T^{Q_P(s,r)} = \prod_{i=1}^{d} \prod_{k=0}^{i-1} e(S_{i-k-1}; \bar{H}_i[j])^{\rho_k}$
  3. Parallel prefix-like, Horner-like on all $S_{i-k-1}^{\rho_k}$



Processor oblivious Parallel Server

degree $d \approx (b \text{ blocks}) \times (q \text{ elements})$

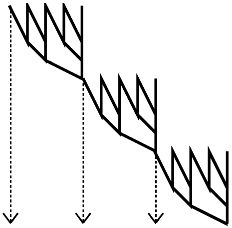
- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations
- Certificate : $\xi = g_T^{Q_P(s,r)} = \prod_{i=1}^{d} \prod_{k=0}^{i-1} e(S_{i-k-1}; \bar{H}_i[j])^{\rho_k}$
  3. Parallel prefix-like, Horner-like on all $S_{i-k-1}^{\rho_k}$



Processor oblivious Parallel Server

degree $d \approx (b \text{ blocks}) \times (q \text{ elements})$

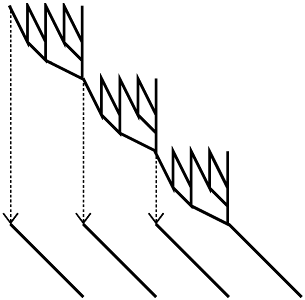
- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations
- Certificate : $\xi = g_T^{Q_P(s,r)} = \prod_{i=1}^{d} \prod_{k=0}^{i-1} e(S_{i-k-1}; \bar{H}_i[j])^{\rho_k}$
  3. Parallel prefix-like, Horner-like on all $S_{i-k-1}^{\rho_k}$



Processor oblivious Parallel Server

degree $d \approx (b$ blocks$) \times (q$ elements$)$

- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations
- Certificate : $\xi = g_T^{Q_P(s,r)} = \prod_{i=1}^{d} \prod_{k=0}^{i-1} e(S_{i-k-1}; \bar{H}_i[j])^{\rho_k}$
  3. Parallel prefix-like, Horner-like on all $S_{i-k-1}^{\rho_k}$



Processor oblivious Parallel Server

degree $d \approx$ (b blocks) $\times$ (q elements)

- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations
- Certificate : $\xi = g_T^{Q_P(s,r)} = \prod_{i=1}^{d} \prod_{k=0}^{i-1} e(S_{i-k-1}; \bar{H}_i[j])^{\rho_k}$
  3. Parallel prefix-like, Horner-like on all $S_{i-k-1}^{\rho_k}$
  4. Parallel blocks of simultaneous pairings
     - **parfor** $k = 1..q$ **do** $\boxed{\bar{\xi}_k[j] \leftarrow \prod_{\ell=b_{k-1}}^{b_k-1} e(u_\ell; \bar{H}_{\ell-1}[j])}$ **endparfor** {q blocks in parallel}
     - Parallel associative reduction: $\boxed{\bar{\xi}[j] \leftarrow \prod_{k=1}^{q} \bar{\xi}_k[j]}$ {$\log_2(q)$ parallel steps}

Processor oblivious Parallel Server

$$\text{degree } d \approx (b \text{ blocks}) \times (q \text{ elements})$$

- Ciphered evaluation : $\zeta = \mathbf{w}^\intercal \odot [r^i]$
  1. Parallel geometric progression
  2. Parallel blocks of simultaneous exponentiations
- Certificate : $\xi = g_T^{Q_P(s,r)} = \prod_{i=1}^{d} \prod_{k=0}^{i-1} e(S_{i-k-1}; \bar{H}_i[j])^{\rho_k}$
  3. Parallel prefix-like, Horner-like on all $S_{i-k-1}^{\rho_k}$
  4. Parallel blocks of simultaneous pairings

VESPo Sequential Performance

- libsnark.git: unciphered, static, circuits verification
- VESPo, open-source C++ Artifact: https://github.com/jgdumas/vespo
  - gmp-6.2.1 & linbox-team/givaro-4.2.0 for modular operations
  - linbox-team/fflas-ffpack-2.5.0 for dense linear algebra
  - relic-0.6.0 for Paillier ($\approx 60\%$) & Pairings ($\approx 40\%$)

254-bits poly. eval.	Client (1 core)	Proof size	Server (1 core)			
			d° 256	1 024	8 192	131 072
Horner (no verif., no crypt.)	-	-	<0.1ms	0.2ms	1.6ms	32.0ms
libsnark (no crypt.)	3.8ms	287B	0.06s	0.20s	1.32s	18.90s
Here (v. & c. & dyn.)	1.6ms	320B	0.21s	0.80s	6.43s	103.07s

Parallel (OpenMP) Server-side VESPo (xeon 6330, @2.00GHz))

$$\zeta = E(P(r)) \qquad \& \qquad \xi = g_T^{Q_P(s,r)}$$

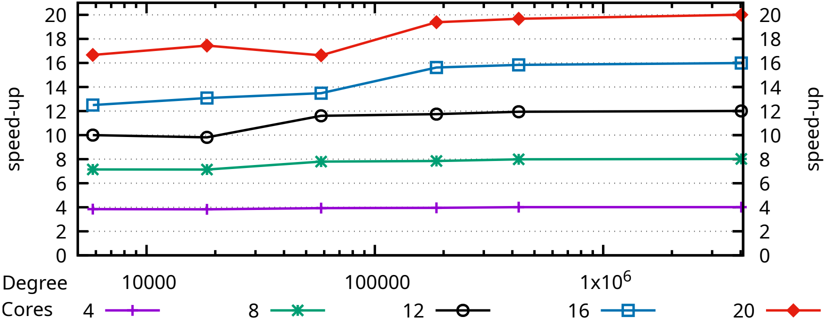


Table: LHE = Paillier-2048: $\zeta \approx 60\%$; Pairing certificate = BN254: $\xi \approx 40\%$

Proof size is 320B; Client verification takes 1.6ms

Parallel (OpenMP) Server-side VESPo (xeon 6330, @2.00GHz))

$$\zeta = E(P(r)) \qquad \& \qquad \xi = g_T^{Q_P(s,r)}$$

Degree	5 816	18 390	58 154	186 093	426 519	4 026 778
1 core	5.0s	15.7s	49.9s	160.9s	373.8s	3 537.5s
4 cores	1.3s	4.1s	12.7s	40.7s	93.2s	881.9s
8 cores	0.7s	2.2s	6.4s	20.5s	46.8s	441.1s
12 cores	0.5s	1.6s	4.3s	13.7s	31.3s	294.6s
16 cores	0.4s	1.2s	3.7s	10.3s	23.6s	221.2s
20 cores	0.3s	0.9s	3.0s	8.3s	19.0s	176.8s

Table: LHE = Paillier-2048: $\zeta \approx 60\%$; Pairing certificate = BN254: $\xi \approx 40\%$

Proof size is 320B; Client verification takes 1.6ms

Protocol 2: DPoR+VESPo

additional Homomorphic routines:

- Pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
- Any linearly homomorphic cryptosystem (LHE): E, D

	Client	Communications	Server
Init	Secrets $\mu, s, \alpha, \beta, \mathbf{\Phi}$ $\mathbf{w}^\intercal = E\left([\mu^i]^\intercal \mathbf{A}\right), \mathcal{K} = g_T^{\bar{P}(s)}$		

Protocol 2: DPoR+VESPo

additional Homomorphic routines:

- Pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$
- Any linearly homomorphic cryptosystem (LHE): E, D

	Client 	Communications	 Server
Init	Secrets $\mu, s, \alpha, \beta, \mathbf{\Phi}$ $\mathbf{w}^\intercal = E\left([\mu^i]^\intercal \mathbf{A}\right), \mathcal{K} = g_T^{\bar{P}(s)}$	$\xrightarrow{\mathbf{A}, \mathbf{w}, S, \bar{H}}$	

Protocol 2: DPoR+VESPo

additional Homomorphic routines:

- Pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
- Any linearly homomorphic cryptosystem (LHE): E, D

	Client	Communications	Server
Init	Secrets $\mu, s, \alpha, \beta, \mathbf{\Phi}$ $\mathbf{w}^\intercal = E\left([\mu^i]^\intercal \mathbf{A}\right), \mathcal{K} = g_T^{\bar{P}(s)}$	$\mathbf{A}, \mathbf{w}, S, \bar{H}$ ⇢	
Audit	Random r $\mathbf{c} = \left((r\mathbf{\Phi})^{d+1} - I_2\right)(r\mathbf{\Phi} - I_2)^{-1}\beta$	r ⇢	

Protocol 2: DPoR+VESPo

additional Homomorphic routines:

- Pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
- Any linearly homomorphic cryptosystem (LHE): E, D

	Client 	Communications	 Server
Init	Secrets $\mu, s, \alpha, \beta, \mathbf{\Phi}$		
	$\mathbf{w}^\intercal = E\left([\mu^i]^\intercal \mathbf{A}\right), \mathcal{K} = g_T^{\bar{P}(s)}$	$\mathbf{A}, \mathbf{w}, S, \bar{H}$ →	
Audit	Random r	r →	$\mathbf{y} = \mathbf{A}[r^i]$
	$\mathbf{c} = \left((r\mathbf{\Phi})^{d+1} - I_2\right)(r\mathbf{\Phi} - I_2)^{-1}\beta$		$\zeta = \mathbf{w}^\intercal \odot [r^i]$
		← $\mathbf{y}, \langle \zeta, \xi \rangle$	$\xi = g_T^{Q_P(s,r)}$

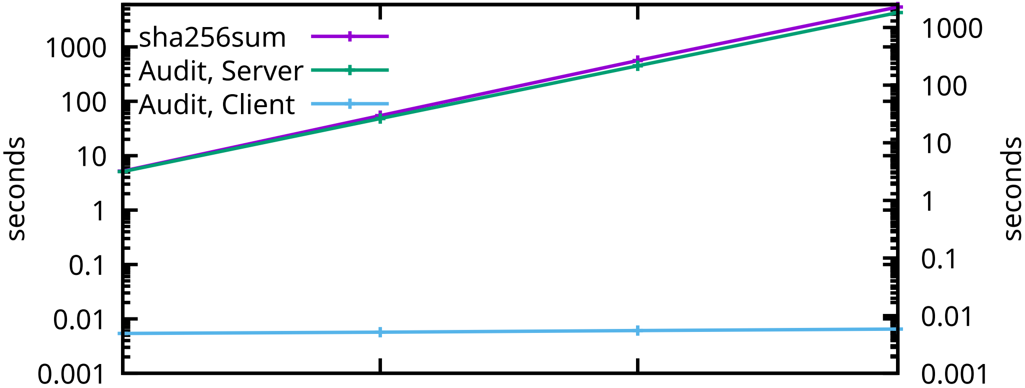
Protocol 2: DPoR+VESPo

additional Homomorphic routines:

- Pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$
- Any linearly homomorphic cryptosystem (LHE): E, D

	Client 	Communications	 Server
Init	Secrets $\mu, s, \alpha, \beta, \mathbf{\Phi}$		
	$\mathbf{w}^\intercal = E\left([\mu^i]^\intercal \mathbf{A}\right), \mathcal{K} = g_T^{\bar{P}(s)}$	$\xrightarrow{\mathbf{A}, \mathbf{w}, S, \bar{H}}$	
Audit	Random r	$\xrightarrow{r}$	$\mathbf{y} = \mathbf{A}[r^i]$
	$\mathbf{c} = \left((r\mathbf{\Phi})^{d+1} - I_2\right)(r\mathbf{\Phi} - I_2)^{-1}\beta$		$\zeta = \mathbf{w}^\intercal \odot [r^i]$
	checks $\mathcal{K} \stackrel{?}{=} \xi^{s-r} g_T^{D(\zeta)\alpha + \mathbf{c}}$	$\xleftarrow{\mathbf{y}, \langle \zeta, \xi \rangle}$	$\xi = g_T^{Q_P(s,r)}$
	checks $D(\zeta) \stackrel{?}{=} [\mu^i]^\intercal \mathbf{y}$		

Protocol 2: DPoR+VESPo (1 core) benchmarks (xeon 6126, @2.60GHz)



Database:	1GB	10GB	100GB	1TB
Proof size:	0.2MB	0.2MB	0.2MB	0.3MB
Client keys:	1KB	1KB	1KB	1KB

Dynamic Proofs of Retrievability

	Client		Audit	Server	
	Storage	Audit Comput.	Comm.	Extra Storage	Audit Comput.
[Shi et al.]	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$	$\mathcal{O}(N)$	$\mathcal{O}(\log N)$
Protocol 1	$\mathcal{O}\left(\sqrt{N}\right)$	$\mathcal{O}\left(\sqrt{N}\right)$	$\mathcal{O}\left(\sqrt{N}\right)$	$o(N)$	$N + o(N)$

Downside: a priori slow $N + o(N)$ server-time for audits.

Dynamic Proofs of Retrievability

	Client 			 Server	
	Storage	Audit Comput.	Audit Comm.	Extra Storage	Audit Comput.
[Shi et al.]	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$	$\mathcal{O}(N)$	$\mathcal{O}(\log N)$
Protocol 1	$\mathcal{O}\left(\sqrt{N}\right)$	$\mathcal{O}\left(\sqrt{N}\right)$	$\mathcal{O}\left(\sqrt{N}\right)$	$o(N)$	$N + o(N)$
Protocol 2 *[VESPo]*	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$	$o(N)$	$N + o(N)$

Downside: a priori slow $N + o(N)$ server-time for audits.

Dynamic Proofs of Retrievability

	Client 			 Server	
	Storage	Audit Comput.	Audit Comm.	Extra Storage	Audit Comput.
[Shi et al.]	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$	$\mathcal{O}(N)$	$\mathcal{O}(\log N)$
Protocol 1	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	$o(N)$	$N + o(N)$
Protocol 2 *[VESPo]*	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$	$\mathcal{O}(\log N)$	$o(N)$	$N + o(N)$

Downside: a priori slow $N + o(N)$ server-time for audits.

But:

- This tradeoff is inherent from our lower bound
- Our Audits are still very inexpensive: 1TB audit on a 4-core VM costs
  - ✓ Example: <5 minutes and \$0.08 USD for 19ms private-verified Protocol 1
- By contrast, storing an extra 1TB on cloud costs from $\approx$**\$50 USD / month**

Outline

Public Auditing

Goal: Let anyone perform an audit

Problem: Audit depends on **client secrets** $\mathbf{u}$, $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$

Public Auditing

Goal: Let anyone perform an audit
Problem: Audit depends on **client secrets** $\mathbf{u}$, $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$

Solution: Use a hash-like function $h(\alpha)$ which is:

- Collision-resistant
- Linearly homomorphic, i.e., $h(\alpha + \beta) = h(\alpha) \oplus h(\beta) \ldots$ (compatible with linear algebra!)

Public Auditing

Goal: Let anyone perform an audit
Problem: Audit depends on **client secrets** $\mathbf{u}$, $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$

Solution: Use a hash-like function $h(\alpha)$ which is:

- Collision-resistant
- Linearly homomorphic, i.e., $h(\alpha + \beta) = h(\alpha) \oplus h(\beta) \ldots$ (compatible with linear algebra!)

We pick $h(\alpha) = g^\alpha$ and completely switch to **computational security**

- g a DLOG-hard elliptic curve group generator
- LIP security assumption (1D *Decision Linear* variant) *[Abdalla et al. Crypto 2015]*

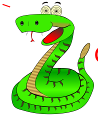
Note: $h(\mathbf{u}) = g^{\mathbf{u}}$ is computed component-wise

Private vs Public Audit

secrets

Private Audit: $(\mathbf{u}^\top \mathbf{A})\,\mathbf{x} = \mathbf{u}^\top\ (\mathbf{B}\mathbf{x})$

Auditor chooses



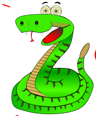
computes $\mathbf{y} = \mathbf{B}\mathbf{x}$

Private vs Public Audit

secrets

Private Audit: $(\mathbf{u}^\intercal \mathbf{A})\,\mathbf{x} = \mathbf{u}^\intercal\,(\mathbf{B}\mathbf{x})$

Auditor chooses



computes $\mathbf{y} = \mathbf{B}\mathbf{x}$

Public Audit: $h(\mathbf{u}^\intercal \mathbf{A})\,\mathbf{x} = h(\mathbf{u}^\intercal)\,(\mathbf{B}\mathbf{x})$

Published

- $\mathbf{K} \leftarrow g^{\mathbf{u}} = h(\mathbf{u})$
- $\mathbf{W} \leftarrow g^{\mathbf{v}} = h(\mathbf{v}) = h(\mathbf{u}^\intercal \mathbf{A})$

⇨ $\mathbf{W}^{\mathbf{x}} \stackrel{?}{=} \mathbf{K}^{\mathbf{y}}$.

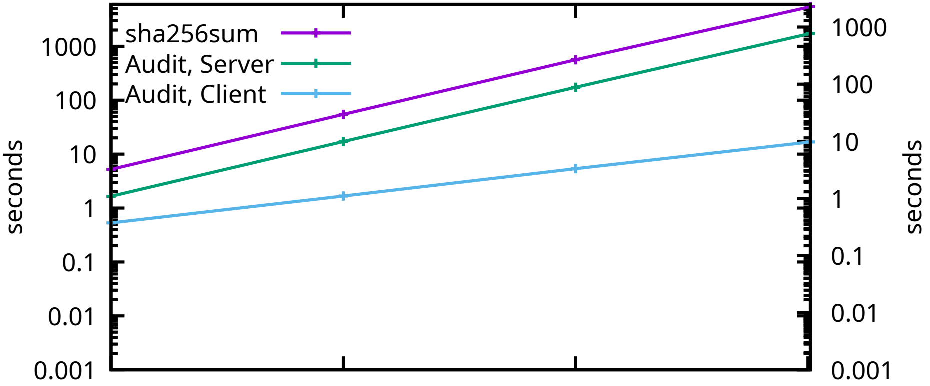
Details of the **Public** Protocol 3

	Client	Communications	Server
Init	$s \xleftarrow{\$} S \subseteq \mathbb{Z}_p$ form $\mathbf{u} = [\mathbf{s}^j]_{j=1\ldots m} \in \mathbb{Z}_p^m$ $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$, $\mathbf{W}^\intercal = g^{\mathbf{v}} \in \mathbb{G}^n$ Publish $r_{\mathbf{A}}, r_{\mathbf{W}}$ and $\mathbf{K} = g^{\mathbf{u}}$	$N = mn \log_2 q$ $\mathbb{G}$ of order p and gen. g $\kappa, \lambda, b, \mathbf{A}, \mathbf{W} \longrightarrow$ **MTInit** $\longrightarrow \mathbf{A}, T_{\mathbf{A}}, \mathbf{W}, T_{\mathbf{W}}$ $r_{\mathbf{A}}, r_{\mathbf{W}} \longleftarrow$	Store $\mathbf{A}, T_{\mathbf{A}}, \mathbf{W}, T_{\mathbf{W}}$
Write	$\mathbf{W}'_j = \mathbf{W}_j \cdot \mathbf{K}_i^{\mathbf{A}'_{ij} - \mathbf{A}_{ij}}$ Update & Publish $r'_{\mathbf{A}}, r'_{\mathbf{W}}$	$i, j, \mathbf{A}'_{ij} \longrightarrow$ **MTVerifiedReads** $\longleftarrow \mathbf{A}, T_{\mathbf{A}}$ $\mathbf{A}_{ij}, \mathbf{W}_j \longleftarrow$ $\longleftarrow \mathbf{W}, T_{\mathbf{W}}$	$\mathbf{W}'_j = \mathbf{W}_j \cdot \mathbf{K}_i^{\mathbf{A}'_{ij} - \mathbf{A}_{ij}}$ Update $\mathbf{A}', T'_{\mathbf{A}}, \mathbf{W}', T'_{\mathbf{W}}$
Audit	$r \xleftarrow{\$} S \subseteq \mathbb{Z}_p^*$ form $\mathbf{x} = [r^i]_{i=1\ldots n} \in \mathbb{Z}_p^n$ $\mathbf{W}^{\mathbf{x}} \stackrel{?}{=} \mathbf{K}^{\mathbf{y}}$	$\xrightarrow{r}$ $\mathbf{W} \longleftarrow$ **MTVerifiedRead** $\longleftarrow \mathbf{W}, T_{\mathbf{W}}$ $\xleftarrow{\mathbf{y}}$	form $\mathbf{x} = [r^i]_{i=1\ldots n} \in \mathbb{Z}_p^n$ $\mathbf{y} = \mathbf{A}\mathbf{x}$

Details of the **Public** Protocol 3

	Client	Communications	Server
Init	$s \xleftarrow{\$} S \subseteq \mathbb{Z}_p$ form $\mathbf{u} = [\mathbf{s}^j]_{j=1\ldots m} \in \mathbb{Z}_p^m$ $\mathbf{v}^\intercal = \mathbf{u}^\intercal \mathbf{A}$, $\mathbf{W}^\intercal = g^{\mathbf{v}} \in \mathbb{G}^n$ Publish $r_{\mathbf{A}}, r_{\mathbf{W}}$ and $\mathbf{K} = g^{\mathbf{u}}$	$N = mn \log_2 q$ $\mathbb{G}$ of order p and gen. g $\kappa, \lambda, b, \mathbf{A}, \mathbf{W} \longrightarrow$ **MTInit** $\longrightarrow \mathbf{A}, T_{\mathbf{A}}, \mathbf{W}, T_{\mathbf{W}}$ $r_{\mathbf{A}}, r_{\mathbf{W}} \longleftarrow$	Store $\mathbf{A}, T_{\mathbf{A}}, \mathbf{W}, T_{\mathbf{W}}$
Write	$\mathbf{W}'_j = \mathbf{W}_j \cdot \mathbf{K}_i^{\mathbf{A}'_{ij} - \mathbf{A}_{ij}}$ Update & Publish $r'_{\mathbf{A}}, r'_{\mathbf{W}}$	$i, j, \mathbf{A}'_{ij} \longrightarrow$ **MTVerifiedReads** $\longleftarrow \mathbf{A}, T_{\mathbf{A}}$ $\mathbf{A}_{ij}, \mathbf{W}_j \longleftarrow$ $\longleftarrow \mathbf{W}, T_{\mathbf{W}}$	$\mathbf{W}'_j = \mathbf{W}_j \cdot \mathbf{K}_i^{\mathbf{A}'_{ij} - \mathbf{A}_{ij}}$ Update $\mathbf{A}', T'_{\mathbf{A}}, \mathbf{W}', T'_{\mathbf{W}}$
Audit	$r \xleftarrow{\$} S \subseteq \mathbb{Z}_p^*$ form $\mathbf{x} = [r^i]_{i=1\ldots n} \in \mathbb{Z}_p^n$ $\mathbf{W}^x \stackrel{?}{=} \mathbf{K}^y$	r $\dashrightarrow$ $\mathbf{W} \longleftarrow$ **MTVerifiedRead** $\longleftarrow \mathbf{W}, T_{\mathbf{W}}$ $\mathbf{y}$ $\dashleftarrow$	form $\mathbf{x} = [r^i]_{i=1\ldots n} \in \mathbb{Z}_p^n$ $\mathbf{y} = \mathbf{A}\mathbf{x}$

Public Audit Compared to MD5 (xeon 6126, @2.60GHz)



Database:	1GB	10GB	100GB	1TB
Proof size:	0.4MB	1.1MB	3.5MB	11.3MB
Client keys:	0.2MB	0.6MB	1.8MB	5.6MB

Outline

Microbenchmarks (xeon 6126, @2.60GHz)

Database	1GB	10GB	100GB	1TB	

Protocol 1: Private audit using 57-bits prime

Matrix view	12339×12432	39131×39200	123831×123872	396281×396368	
Server extra storage	<0.01%	<0.01%	<0.01%	<0.01%	$o(N)$
Client Storage (keys)	169KB	535KB	1 693KB	5 418KB	$O(\sqrt{N})$
Server Audit (1 | 12 cores)	0.29s | 0.04s	2.68s | 0.30s	29.04s | 3.36s	219.7s | 41.48s	$O(N)$
Communications (proof size)	169KB	535KB	1 693KB	5 418KB	$O(\sqrt{N})$
Client Audit (1 core)	0.6ms	1.7ms	5.3ms	18.3ms	$O(\sqrt{N})$

Protocol 2: Private Rectangular Dynamic-ciphered delegated polynomial evaluation with 254-bits groups

Matrix view	6599×5125	7265×46551	7929×426519	8600×4026778	
Server extra storage	0.11%	0.10%	0.09%	0.08%	$o(N)$
Client storage (keys)	0.94KB	0.94KB	0.94KB	0.94KB	$O(1)$
Server Audit (1 | 12 cores): matrix-vector step	1.1s | 0.2s	11.3s | 1.3s	113.2s | 12.8s	1 147.9s | 130.7s	$O(N)$
Server Audit (1 | 12 cores): polynomial step	3.8s | 0.4s	35.5s | 3.6s	324.1s | 30.6s	3 064.8s | 283.6s	$o(N)$
Communications (proof size)	205KB	226KB	246KB	267KB	$O(\log N)$
Client Audit (1 core): dotproduct step	3.7ms	4.0ms	4.4ms	4.8ms	$O(\log N)$
Client Audit (1 core): polynomial step	1.7ms	1.7ms	1.7ms	1.7ms	

Transatlantic Audit times & costs (n1-standard)

cores	Metric	1GB	10GB	100GB	1TB
	regional monthly	\$0.09	\$0.89	\$8.80	\$90.11

Protocol 1 Private-verified audit using **57-bit** prime

cores	Metric	1GB	10GB	100GB	1TB
1	Client Audit	0.000 2s	0.000 5s	0.0076s	0.0188s
4	Server Audit	0.06s	0.62s	29.08s	278.37s
	Cost	\$0.000 02	\$0.000 2	\$0.008	\$0.080
16	Server Audit	0.03s	0.22s	1.88s	250.91s
	Cost	\$0.000 02	\$0.000 2	\$0.001	\$0.175

Protocol 3 Public-verified audit using **ristretto255**

cores	Metric	1GB	10GB	100GB	1TB
1	Client Audit	0.5s	1.7s	5.4s	16.8s
4	Server Audit	0.45s	4.37s	51.45s	536.09s
	Cost	\$0.000 1	\$0.001	\$0.015	\$0.155
16	Server Audit	0.12s	1.21s	11.87s	357.49s
	Cost	\$0.000 1	\$0.001	\$0.008	\$0.249

Summary

Our new DPoR provides:

- ✓ Fast reads/updates
- ✓ Transparent and small server storage
- ✓ Provable retrievability after successful audits
- ✓ Sub-linear `Audit` bandwidth and client time
- ✓ A public-verifiable variant

Also novel:

- ✓ Efficient & Verified evaluation of, secret & dynamic, polynomials

Open:

- ✗ Efficient & **Publicly** verified evaluation of, secret & dynamic, polynomials

Thank you

Thank you!