Approximate Lower Bound Arguments (ALBA)

Pyrros Chaidos[1,4] Aggelos Kiayias[2,4] Leo Reyzin[3] Tolik Zinovyev[3]

[1]National & Kapodistrian University of Athens

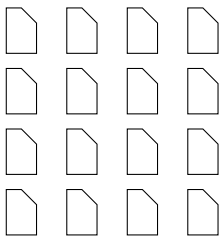
[2]University of Edinburgh

[3]Boston University

[4]IOG

Problem Statement

Prover

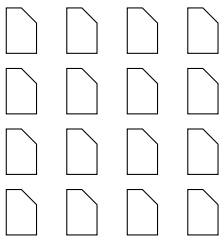


$R(\square) = 1$

Problem Statement

goal: "I know $> n_{\mathrm{f}}$ elements with $R(\cdot) = 1$"

Prover ⟶ Verifier

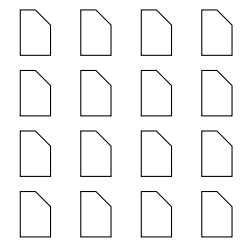


$R($ $) = 1$

Problem Statement

goal: "I know $> n_{\mathrm{f}}$ elements with $R(\cdot) = 1$"

Prover ⟶ Verifier



n_{p} $(n_{\mathrm{p}} > n_{\mathrm{f}})$

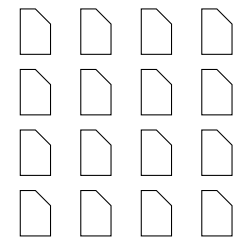
$R($  $) = 1$

Problem Statement

goal: "I know $> n_{\mathrm{f}}$ elements  with $R(\cdot) = 1$"

how: send some  with special properties

Prover $\longrightarrow$ Verifier



n_{p} ($n_{\mathrm{p}} > n_{\mathrm{f}}$)

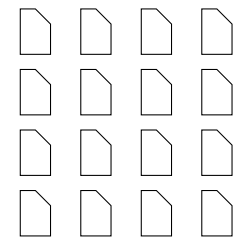
$R($  $) = 1$

Problem Statement

goal: "I know $> n_{\mathrm{f}}$ elements 🗋 with $R(\cdot) = 1$"

how: send some 🗋 with special properties

Prover $\longrightarrow$ Verifier



n_{p} ($n_{\mathrm{p}} > n_{\mathrm{f}}$)

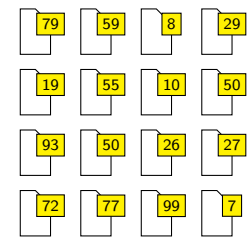
$R(\text{🗋}) = 1$

Larger ratio $n_{\mathrm{p}}/n_{\mathrm{f}}$ enables more efficient schemes.

Problem Statement

goal: "I know $> n_{\mathrm{f}}$ elements 🗋 with $R(\cdot) = 1$"
how: send some 🗋 with special properties

Prover ⟶ Verifier



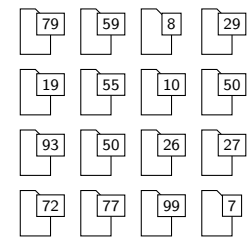
$R(🗋) = 1$

Larger ratio $n_{\mathrm{p}}/n_{\mathrm{f}}$ enables more efficient schemes.

Problem Statement

goal: "I know $> n_\mathrm{f}$ elements 🗋 with $R(\cdot) = 1$"
how: send some 🗋 with special properties

Prover ⟶ Verifier



~~$R(🗋) = 1$~~

$W(🗋) \in \mathbb{N}$

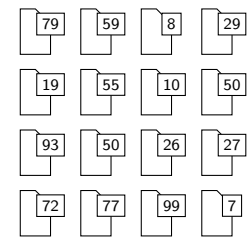
Larger ratio $n_\mathrm{p}/n_\mathrm{f}$ enables more efficient schemes.

Problem Statement

general goal: "I know 🗋 with $\sum W(\cdot) > n_{\mathrm{f}}$"

how: send some 🗋 with special properties

Prover $\longrightarrow$ Verifier



$W(🗋) \in \mathbb{N}$

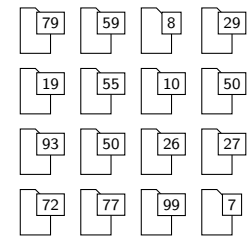
Larger ratio $n_{\mathrm{p}}/n_{\mathrm{f}}$ enables more efficient schemes.

Problem Statement

general goal: "I know 🗋 with $\sum W(\cdot) > n_{\mathrm{f}}$"

how: send some 🗋 with special properties

Prover ⟶ Verifier



$W(🗋) \in \mathbb{N}$

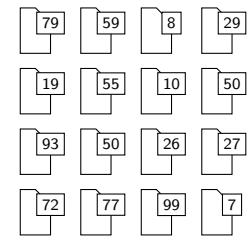
$\sum W(\cdot) = n_{\mathrm{p}}$

Larger ratio $n_{\mathrm{p}}/n_{\mathrm{f}}$ enables more efficient schemes.

Problem Statement

general goal: "I know 🗋 with $\sum W(\cdot) > n_{\mathrm{f}}$"
how: send some 🗋 with special properties

Prover $\longrightarrow$ Verifier

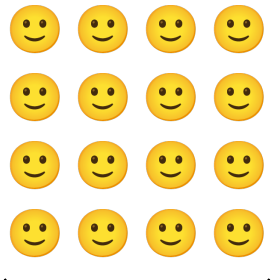


$W(🗋) \in \mathbb{N}$

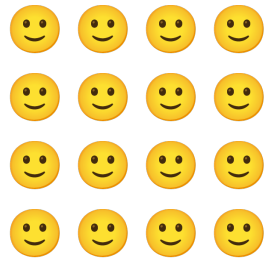
Larger ratio $n_{\mathrm{p}}/n_{\mathrm{f}}$ enables more efficient schemes.

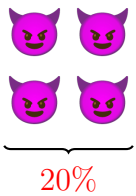
"Approximate Lower Bound Argument" (ALBA)

Example: Consensus



Example: Consensus





$$n_{\mathrm{p}}/n_{\mathrm{f}} = 4$$

History

1985 "Approximate lower bound" term appears first in László Babai's work

1983 Sipser and Gács use an ALBA protocol to prove $\mathrm{BPP} \subseteq \mathrm{RP}^{\mathrm{NP}}$

1986 Goldwasser and Sipser use an ALBA protocol to prove $\mathrm{IP}[Q] \subseteq \mathrm{AM}[Q+2]$

Above implementations of ALBA have interaction and large proofs.

Our goal – ALBA for real world applications:

- minimize proof size
- non-interactive
- fast prover & verifier
- minimize communication in decentralized setting
- extractability (proof of knowledge)

History

1985 "Approximate lower bound" term appears first in László Babai's work

1983 Sipser and Gács use an ALBA protocol to prove $\mathrm{BPP} \subseteq \mathrm{RP}^{\mathrm{NP}}$

1986 Goldwasser and Sipser use an ALBA protocol to prove $\mathrm{IP}[Q] \subseteq \mathrm{AM}[Q+2]$

Above implementations of ALBA have interaction and large proofs.

Our goal – ALBA for real world applications:

- minimize proof size
- non-interactive
- fast prover & verifier
- minimize communication in decentralized setting
- extractability (proof of knowledge)

History

1985 "Approximate lower bound" term appears first in László Babai's work

1983 Sipser and Gács use an ALBA protocol to prove $\mathrm{BPP} \subseteq \mathrm{RP}^{\mathrm{NP}}$

1986 Goldwasser and Sipser use an ALBA protocol to prove $\mathrm{IP}[Q] \subseteq \mathrm{AM}[Q+2]$

Above implementations of ALBA have interaction and large proofs.

Our goal – ALBA for real world applications:

- minimize proof size
- non-interactive
- fast prover & verifier
- minimize communication in decentralized setting
- extractability (proof of knowledge)

Defining ALBA

- Non-interactive proof system in either
  - random oracle (RO) model:
    prover and verifier have oracle access to random $H : \{0,1\}^* \to \{0,1\}^k$
  - common reference string (CRS) model:
    prover and verifier are given $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$
- Prover and verifier have oracle access to weight function W
- Completeness: if prover has elements S_p of total weight $n_\mathrm{p} \implies$ prob. $1 - \mathrm{negl}$
  - $\pi \leftarrow \mathsf{Prove}^{H,W}(S_\mathrm{p})$
  - $\mathsf{Verify}^{H,W}(\pi) = 1$
- Soundness: if $\mathcal{A}$ makes valid proof $\implies$ extract elements of total weight $> n_\mathrm{f}$; negligible knowledge error
  - $S_\mathrm{f} \leftarrow \mathsf{Extract}^{H,W,\mathcal{A}}()$

Defining ALBA

- Non-interactive proof system in either
  - random oracle (RO) model:
    prover and verifier have oracle access to random $H : \{0,1\}^* \to \{0,1\}^k$
  - common reference string (CRS) model:
    prover and verifier are given $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$
- Prover and verifier have oracle access to weight function W
- Completeness: if prover has elements S_p of total weight $n_\mathrm{p} \implies$ prob. $1 - \mathrm{negl}$
  - $\pi \leftarrow \mathsf{Prove}^{H,W}(S_\mathrm{p})$
  - $\mathsf{Verify}^{H,W}(\pi) = 1$
- Soundness: if $\mathcal{A}$ makes valid proof $\implies$ extract elements of total weight $> n_\mathrm{f}$; negligible knowledge error
  - $S_\mathrm{f} \leftarrow \mathsf{Extract}^{H,W,\mathcal{A}}()$

Defining ALBA

- Non-interactive proof system in either
  - random oracle (RO) model:
    prover and verifier have oracle access to random $H : \{0,1\}^* \to \{0,1\}^k$
  - common reference string (CRS) model:
    prover and verifier are given $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$
- Prover and verifier have oracle access to weight function W
- Completeness: if prover has elements S_p of total weight $n_\mathrm{p} \implies$ prob. $1 - \mathrm{negl}$
  - $\pi \leftarrow \mathsf{Prove}^{H,W}(S_\mathrm{p})$
  - $\mathsf{Verify}^{H,W}(\pi) = 1$
- Soundness: if $\mathcal{A}$ makes valid proof $\implies$ extract elements of total weight $> n_\mathrm{f}$; negligible knowledge error
  - $S_\mathrm{f} \leftarrow \mathsf{Extract}^{H,W,\mathcal{A}}()$

Defining ALBA

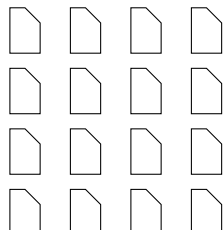
- Non-interactive proof system in either
  - random oracle (RO) model:
    prover and verifier have oracle access to random $H : \{0,1\}^* \to \{0,1\}^k$
  - common reference string (CRS) model:
    prover and verifier are given $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$
- Prover and verifier have oracle access to weight function W
- Completeness: if prover has elements S_p of total weight $n_\mathrm{p} \implies$ prob. $1 - \mathrm{negl}$
  - $\pi \leftarrow \mathsf{Prove}^{H,W}(S_\mathrm{p})$
  - $\mathsf{Verify}^{H,W}(\pi) = 1$
- Soundness: if $\mathcal{A}$ makes valid proof $\implies$ extract elements of total weight $> n_\mathrm{f}$; negligible knowledge error
  - $S_\mathrm{f} \leftarrow \mathsf{Extract}^{H,W,\mathcal{A}}()$

Defining ALBA

- Non-interactive proof system in either
  - random oracle (RO) model:
    prover and verifier have oracle access to random $H : \{0,1\}^* \to \{0,1\}^k$
  - common reference string (CRS) model:
    prover and verifier are given $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$
- Prover and verifier have oracle access to weight function W
- Completeness: if prover has elements S_p of total weight $n_\mathrm{p} \implies$ prob. $1 - \mathrm{negl}$
  - $\pi \leftarrow \mathsf{Prove}^W(\mathrm{crs}, S_\mathrm{p})$
  - $\mathsf{Verify}^W(\mathrm{crs}, \pi) = 1$
- Soundness: if $\mathcal{A}$ makes valid proof $\implies$ extract elements of total weight $> n_\mathrm{f}$; negligible knowledge error
  - $S_\mathrm{f} \leftarrow \mathsf{Extract}^W(\mathcal{A})$

Applications

- Weighted Multisignatures
  - Prove that sufficiently many parties signed a message
  - Proof-of-stake blockchains: prove that parties with sufficient total stake attested to blockchain state
- Straight-Line Witness Extraction for SNARKs
  - Straight-line extraction (no rewinding) useful for composability
- Details to follow



Applications

- Weighted Multisignatures
  - Prove that sufficiently many parties signed a message
  - Proof-of-stake blockchains: prove that parties with sufficient total stake attested to blockchain state
- Straight-Line Witness Extraction for SNARKs
  - Straight-line extraction (no rewinding) useful for composability
- Details to follow

Applications

- Weighted Multisignatures
  - Prove that sufficiently many parties signed a message
  - Proof-of-stake blockchains: prove that parties with sufficient total stake attested to blockchain state
- Straight-Line Witness Extraction for SNARKs
  - Straight-line extraction (no rewinding) useful for composability
- Details to follow

Why not use SNARK to implement ALBA?

- Not possible when the weight function cannot be represented by a single program
- Too costly when number of elements is large
- Instead, use ALBA + SNARK
- Our scheme works in CRS model $\Longrightarrow$ avoid instantiating random oracle inside a circuit

SNARK

"know elements $x_1, ..., x_k$ of total weight $> n_f$"
(witness $x_1, ..., x_k$)

Why not use SNARK to implement ALBA?

- Not possible when the weight function cannot be represented by a single program
- Too costly when number of elements is large
- Instead, use ALBA + SNARK
- Our scheme works in CRS model $\Longrightarrow$ avoid instantiating random oracle inside a circuit

SNARK

"know elements $x_1, ..., x_k$ of total weight $> n_f$"
(witness $x_1, ..., x_k$)

Why not use SNARK to implement ALBA?

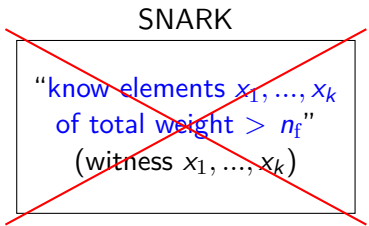
- Not possible when the weight function cannot be represented by a single program
- Too costly when number of elements is large
- Instead, use ALBA + SNARK
- Our scheme works in CRS model $\Longrightarrow$ avoid instantiating random oracle inside a circuit

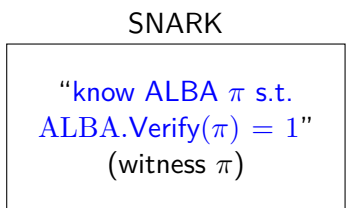
SNARK

"know elements $x_1, ..., x_k$ of total weight $> n_f$"
(witness $x_1, ..., x_k$)

Why not use SNARK to implement ALBA?

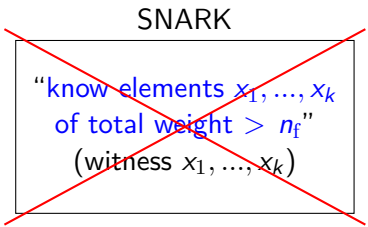
- Not possible when the weight function cannot be represented by a single program
- Too costly when number of elements is large
- Instead, use ALBA + SNARK
- Our scheme works in CRS model $\Longrightarrow$ avoid instantiating random oracle inside a circuit

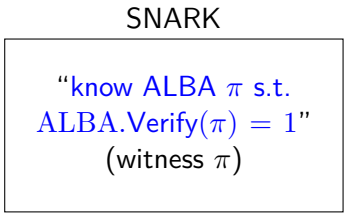




Why not use SNARK to implement ALBA?

- Not possible when the weight function cannot be represented by a single program
- Too costly when number of elements is large
- Instead, use ALBA + SNARK
- Our scheme works in CRS model $\Longrightarrow$ avoid instantiating random oracle inside a circuit





Our Results

Completeness and soundness error $2^{-\lambda}$, $n_{\mathrm{p}} = 2 \cdot n_{\mathrm{f}}$:

proof size (elements)	prover runtime	verifier runtime
$\lambda + \log \lambda + 6$	expected $O(n + \lambda^2)$	$O(\lambda)$

Table: Upper bound

Our Results

Completeness and soundness error $2^{-\lambda}$, $n_{\mathrm{p}} = 2 \cdot n_{\mathrm{f}}$:

proof size (elements)	prover runtime	verifier runtime
141 for $\lambda = 128$	expected $O(n + \lambda^2)$	$O(\lambda)$

Table: Upper bound

Our Results

Completeness and soundness error $2^{-\lambda}$, $n_{\mathrm{p}} = 2 \cdot n_{\mathrm{f}}$:

proof size (elements)	prover runtime	verifier runtime
$\lambda + \log \lambda + 6$	expected $O(n + \lambda^2)$	$O(\lambda)$

Table: Upper bound

Our Results

Completeness and soundness error $2^{-\lambda}$, $n_{\mathrm{p}} = 2 \cdot n_{\mathrm{f}}$:

proof size (elements)	prover runtime	verifier runtime
$\lambda + \log \lambda + 6$	expected $O(n + \lambda^2)$	$O(\lambda)$

Table: Upper bound

proof size (elements)
$> \lambda - 3$

Table: Lower bound

Our Results

Completeness and soundness error $2^{-\lambda}$, ~~$n_{\mathrm{p}} = 2 \cdot n_{\mathrm{f}}$~~:

proof size (elements)	prover runtime	verifier runtime
$\frac{\lambda + \log \lambda + 6}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$	expected $O(n + \lambda^2)$	$O(\lambda)$

Table: Upper bound

proof size (elements)
$> \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$

Table: Lower bound

For simplicity...

- Most constructions unweighted
  - assume honest prover has n_{p} elements
- Simple soundness
  - no proof of knowledge
  - assume malicious prover has n_{f} fixed elements $\Longrightarrow$ succeeds with negligible probability

Coming next...

- ☑ Definitions
- Unweighted case
  - ☐ **Prover-inefficient construction**
  - ☐ Telescope construction
  - ☐ Improving prover running time
  - ☐ Decentralized setting
- ☐ Adding weights
- ☐ Applications

Warmup: Prover-inefficient Construction

- Let proof size be u elements (u to be chosen later)
- Consider all u-sequences of elements
- Honest prover has ${n_\mathrm{p}}^u$ sequences
- Malicious prover has ${n_\mathrm{f}}^u$ sequences
- Notice exponential gap $\left(\frac{n_\mathrm{p}}{n_\mathrm{f}}\right)^u$
- A valid proof will be a sequence chosen with probability ε using RO

Valid proof: $(s_1, ..., s_u)$ s.t.
- $H(s_1, ..., s_u) = 1$;

$(\Pr[H(\cdot) = 1] = \varepsilon)$

Warmup: Prover-inefficient Construction

- Let proof size be u elements (u to be chosen later)
- Consider all u-sequences of elements
- Honest prover has $n_{\mathrm{p}}{}^{u}$ sequences
- Malicious prover has $n_{\mathrm{f}}{}^{u}$ sequences
- Notice exponential gap $\left(\frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}\right)^{u}$
- A valid proof will be a sequence chosen with probability ε using RO

Valid proof: $(s_1, ..., s_u)$ s.t.
- $H(s_1, ..., s_u) = 1$;

$(\Pr[H(\cdot) = 1] = \varepsilon)$

Warmup: Prover-inefficient Construction

- Let proof size be u elements (u to be chosen later)
- Consider all u-sequences of elements
- Honest prover has $n_{\mathrm{p}}{}^{u}$ sequences
- Malicious prover has $n_{\mathrm{f}}{}^{u}$ sequences
- Notice exponential gap $\left(\frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}\right)^{u}$
- A valid proof will be a sequence chosen with probability ε using RO

Valid proof: $(s_1, ..., s_u)$ s.t.
- $H(s_1, ..., s_u) = 1$;

$(\Pr[H(\cdot) = 1] = \varepsilon)$

Warmup: Prover-inefficient Construction

- Let proof size be u elements (u to be chosen later)
- Consider all u-sequences of elements
- Honest prover has ${n_{\mathrm{p}}}^u$ sequences
- Malicious prover has ${n_{\mathrm{f}}}^u$ sequences
- Notice exponential gap $\left(\frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}\right)^u$
- A valid proof will be a sequence chosen with probability ε using RO

Valid proof: $(s_1, ..., s_u)$ s.t.
- $H(s_1, ..., s_u) = 1$;

$(\Pr[H(\cdot) = 1] = \varepsilon)$

Warmup: Prover-inefficient Construction

- Soundness error bounded using union bound: "number" $\cdot$ "probability" $= {n_\mathrm{f}}^u \cdot \varepsilon$
- Completeness error bounded by $(1-\varepsilon)^{{n_\mathrm{p}}^u} \leq e^{-\varepsilon {n_\mathrm{p}}^u}$
- Setting completeness and soundness errors $\leq 2^{-\lambda}$ and calibrating ε, we find $u = \frac{\lambda + \log \lambda}{\log(n_\mathrm{p}/n_\mathrm{f})}$ suffices (lower bound: $u > \frac{\lambda - 3}{\log(n_\mathrm{p}/n_\mathrm{f})}$)
- Exponential prover running time

Valid proof: $(s_1, ..., s_u)$ s.t.
- $H(s_1, ..., s_u) = 1$;

$(\Pr[H(\cdot) = 1] = \varepsilon)$

Warmup: Prover-inefficient Construction

- Soundness error bounded using union bound: "number" · "probability" $= {n_\mathrm{f}}^u \cdot \varepsilon$
- Completeness error bounded by $(1-\varepsilon)^{{n_\mathrm{p}}^u} \leq e^{-\varepsilon {n_\mathrm{p}}^u}$
- Setting completeness and soundness errors $\leq 2^{-\lambda}$ and calibrating ε, we find $u = \frac{\lambda + \log \lambda}{\log(n_\mathrm{p}/n_\mathrm{f})}$ suffices (lower bound: $u > \frac{\lambda - 3}{\log(n_\mathrm{p}/n_\mathrm{f})}$)
- Exponential prover running time

Valid proof: $(s_1, ..., s_u)$ s.t.
- $H(s_1, ..., s_u) = 1$;

$(\Pr[H(\cdot) = 1] = \varepsilon)$

Warmup: Prover-inefficient Construction

- Soundness error bounded using union bound: "number" $\cdot$ "probability" $= {n_\mathrm{f}}^u \cdot \varepsilon$
- Completeness error bounded by $(1-\varepsilon)^{{n_\mathrm{p}}^u} \leq e^{-\varepsilon {n_\mathrm{p}}^u}$
- Setting completeness and soundness errors $\leq 2^{-\lambda}$ and calibrating ε, we find $u = \frac{\lambda + \log \lambda}{\log(n_\mathrm{p}/n_\mathrm{f})}$ suffices (lower bound: $u > \frac{\lambda - 3}{\log(n_\mathrm{p}/n_\mathrm{f})}$)
- Exponential prover running time

Valid proof: $(s_1, ..., s_u)$ s.t.
- $H(s_1, ..., s_u) = 1$;

$(\Pr[H(\cdot) = 1] = \varepsilon)$

Coming next...

- ☑ Definitions
- Unweighted case
  - ☑ Prover-inefficient construction
  - ☐ **Telescope construction**
  - ☐ Improving prover running time
  - ☐ Decentralized setting
- ☐ Adding weights
- ☐ Applications

Efficient Construction: Telescope

- Exploit the gap $\left(\frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}\right)^{u}$ in an efficient way
- Grow sequences of increasing length but keep the number small ("Telescope" method)

Efficient Construction: Telescope

- Exploit the gap $\left(\frac{n_{\mathrm{p}}}{n_{\mathrm{f}}}\right)^{u}$ in an efficient way
- Grow sequences of increasing length but keep the number small ("Telescope" method)

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. ($n = n_{\mathrm{p}}$)

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. ($n = n_{\mathrm{p}}$)

① $\quad$ $\#$

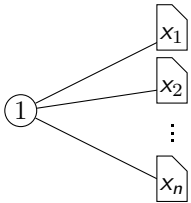
② $\quad$ d

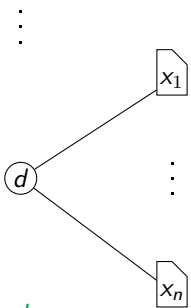
$\vdots$

ⓓ

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. ($n = n_{\mathrm{p}}$)

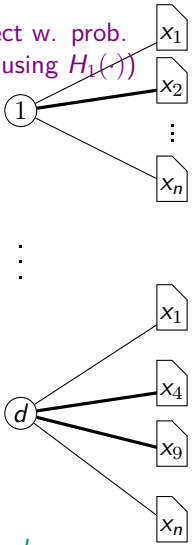




\# d

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. ($n = n_{\mathrm{p}}$)

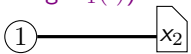


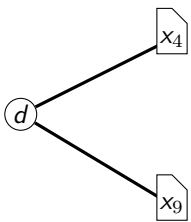
d

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. $(n = n_{\mathrm{p}})$

(select w. prob. $1/n$ using $H_1(\cdot)$)

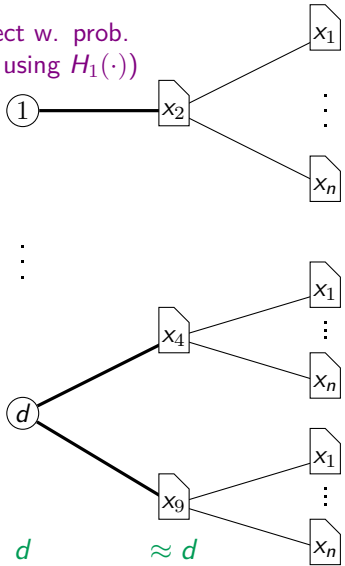




$\#$ d $\approx d$

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. ($n = n_{\mathrm{p}}$)

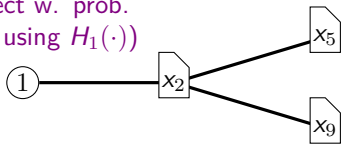


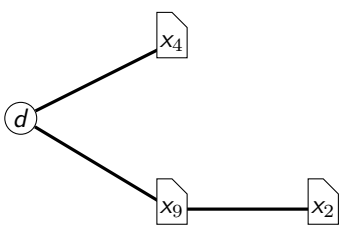
d $\approx d$

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. $(n = n_{\mathrm{p}})$

(select w. prob. $1/n$ using $H_1(\cdot)$)



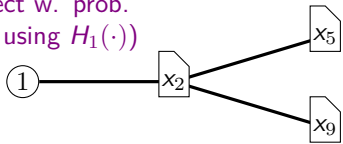


#	d	$\approx d$	$\approx d$

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. $(n = n_{\mathrm{p}})$

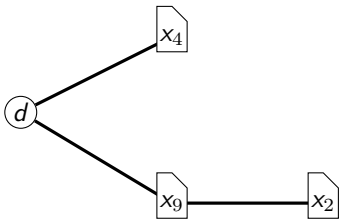
(select w. prob. $1/n$ using $H_1(\cdot)$)



$\vdots$

$\ldots$

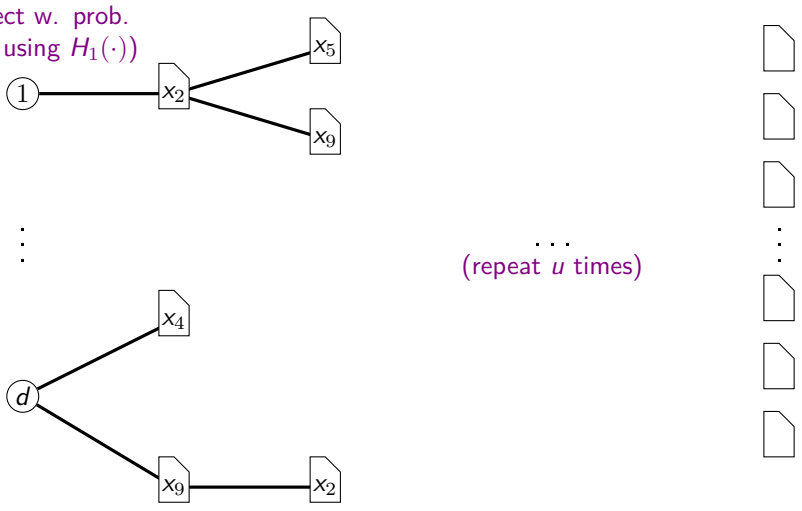
(repeat u times)



#	d	$\approx d$	$\approx d$

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. ($n = n_{\mathrm{p}}$)

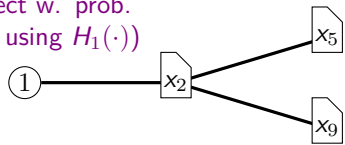


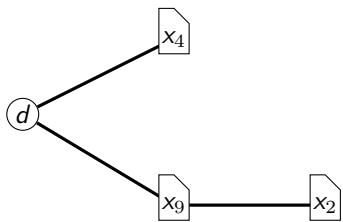
#	d	$\approx d$	$\approx d$	$\approx d$

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. $(n = n_{\mathrm{p}})$

(select w. prob. $1/n$ using $H_1(\cdot)$)





#	d	$\approx d$	$\approx d$

(repeat u times)

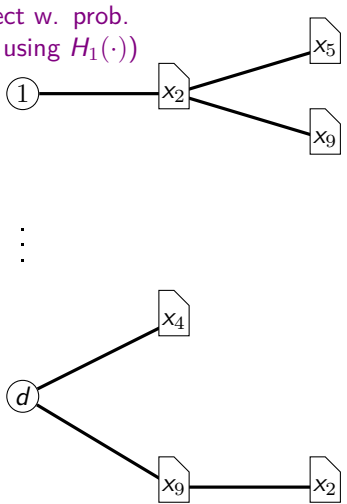
Valid proof: $(t, s_1, ..., s_u)$ s.t.

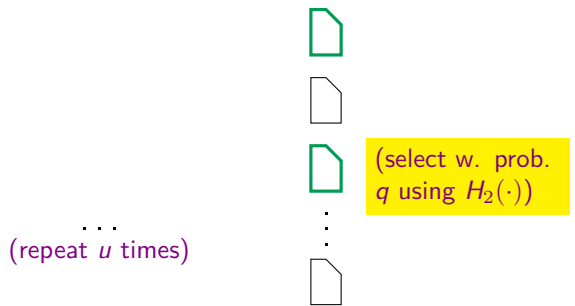
- $1 \leq t \leq d$;
- $\forall 1 \leq i \leq u,\ H_1(t, s_1, ..., s_i) = 1$;

$(\Pr[H_1(\cdot) = 1] = 1/n)$

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. $(n = n_{\mathrm{p}})$





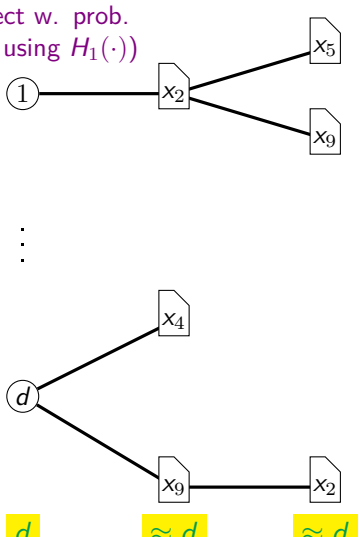
Valid proof: $(t, s_1, ..., s_u)$ s.t.

- $1 \leq t \leq d$;
- $\forall 1 \leq i \leq u,\ H_1(t, s_1, ..., s_i) = 1$;
- $H_2(t, s_1, ..., s_u) = 1$;

$(\Pr[H_1(\cdot) = 1] = 1/n,\ \Pr[H_2(\cdot) = 1] = q)$

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. $(n = n_{\mathrm{p}})$



Valid proof: $(t, s_1, ..., s_u)$ s.t.

- $1 \leq t \leq d$;
- $\forall 1 \leq i \leq u,\ H_1(t, s_1, ..., s_i) = 1$;
- $H_2(t, s_1, ..., s_u) = 1$;

$\left(\Pr[H_1(\cdot) = 1] = 1/n,\ \Pr[H_2(\cdot) = 1] = q\right)$

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. $(n = n_{\mathrm{p}})$

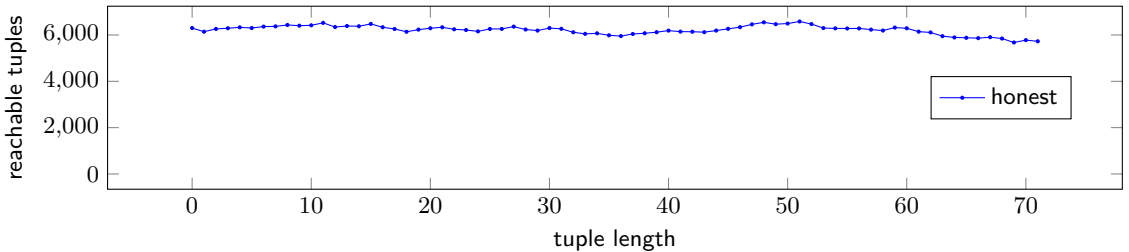


Figure: Telescope simulation: $d = 6300$, $n_{\mathrm{p}} = 1000$, $n_{\mathrm{f}} = 500$

#	d	$\approx d$	$\approx d$

Valid proof: $(t, s_1, ..., s_u)$ s.t.

- $1 \leq t \leq d$;
- $\forall 1 \leq i \leq u,\ H_1(t, s_1, ..., s_i) = 1$;
- $H_2(t, s_1, ..., s_u) = 1$;

$(\Pr[H_1(\cdot) = 1] = 1/n,\ \Pr[H_2(\cdot) = 1] = q)$

Telescope Construction (cont.)

Let $x_1, ..., x_n$ be honest prover's elements. $(n = n_{\mathrm{p}})$

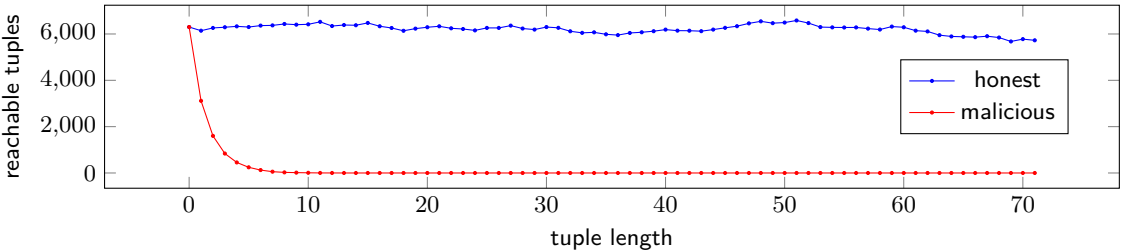


Figure: Telescope simulation: $d = 6300$, $n_{\mathrm{p}} = 1000$, $n_{\mathrm{f}} = 500$

#	d	$\approx d$	$\approx d$

Valid proof: $(t, s_1, ..., s_u)$ s.t.

- $1 \leq t \leq d$;
- $\forall 1 \leq i \leq u,\ H_1(t, s_1, ..., s_i) = 1$;
- $H_2(t, s_1, ..., s_u) = 1$;

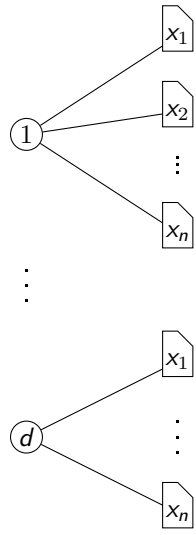
$(\Pr[H_1(\cdot) = 1] = 1/n,\ \Pr[H_2(\cdot) = 1] = q\,)$

Efficient Construction: Telescope (cont.)

- Represent as d trees
- Prover runs DFS (instead of BFS)
- Union bound to analyze soundness:
  - soundness error

    $\text{“number”} \cdot \text{“probability”} = d \cdot n_{\mathrm{f}}^{u} \cdot \left(\frac{1}{n_{\mathrm{p}}}\right)^{u} \cdot q$
- Recursive formula to analyze completeness:
  - $f(0) = 1 - q$
  - $f(k+1) = \left((1 - \frac{1}{n_{\mathrm{p}}}) + \frac{1}{n_{\mathrm{p}}} \cdot f(k)\right)^{n_{\mathrm{p}}}$
  - completeness error $\left(f(u)\right)^{d}$
- Setting errors $\leq 2^{-\lambda}$, achieve proof size $u = \frac{\lambda + \log \lambda + 2}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$

  (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)

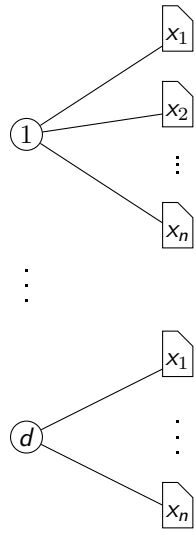


Efficient Construction: Telescope (cont.)

- Represent as d trees
- Prover runs DFS (instead of BFS)
- Union bound to analyze soundness:
  - soundness error

    $\text{“number”} \cdot \text{“probability”} = d \cdot n_{\mathrm{f}}^{u} \cdot \left(\frac{1}{n_{\mathrm{p}}}\right)^{u} \cdot q$
- Recursive formula to analyze completeness:
  - $f(0) = 1 - q$
  - $f(k+1) = \left((1 - \frac{1}{n_{\mathrm{p}}}) + \frac{1}{n_{\mathrm{p}}} \cdot f(k)\right)^{n_{\mathrm{p}}}$
  - completeness error $\left(f(u)\right)^{d}$
- Setting errors $\leq 2^{-\lambda}$, achieve proof size $u = \frac{\lambda + \log \lambda + 2}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$

  (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)

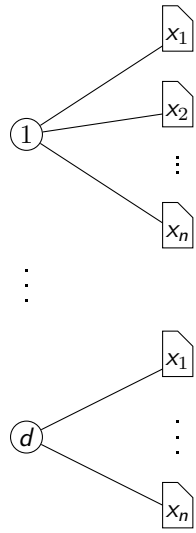


Efficient Construction: Telescope (cont.)

- Represent as d trees
- Prover runs DFS (instead of BFS)
- Union bound to analyze soundness:
  - soundness error

    $\text{“number”} \cdot \text{“probability”} = d \cdot n_{\mathrm{f}}^{u} \cdot \left(\frac{1}{n_{\mathrm{p}}}\right)^{u} \cdot q$
- Recursive formula to analyze completeness:
  - $f(0) = 1 - q$
  - $f(k+1) = \left((1 - \frac{1}{n_{\mathrm{p}}}) + \frac{1}{n_{\mathrm{p}}} \cdot f(k)\right)^{n_{\mathrm{p}}}$
  - completeness error $\left(f(u)\right)^{d}$
- Setting errors $\leq 2^{-\lambda}$, achieve proof size $u = \frac{\lambda + \log \lambda + 2}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$

  (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)

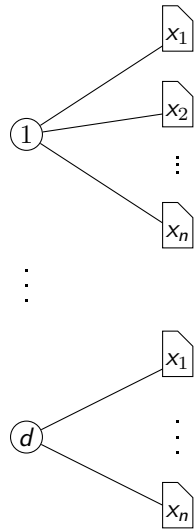


Efficient Construction: Telescope (cont.)

- Represent as d trees
- Prover runs DFS (instead of BFS)
- Union bound to analyze soundness:
  - soundness error

    $\text{“number”} \cdot \text{“probability”} = d \cdot n_{\mathrm{f}}^{u} \cdot \left(\frac{1}{n_{\mathrm{p}}}\right)^{u} \cdot q$
- Recursive formula to analyze completeness:
  - $f(0) = 1 - q$
  - $f(k+1) = \left((1 - \frac{1}{n_{\mathrm{p}}}) + \frac{1}{n_{\mathrm{p}}} \cdot f(k)\right)^{n_{\mathrm{p}}}$
  - completeness error $\left(f(u)\right)^{d}$
- Setting errors $\leq 2^{-\lambda}$, achieve proof size $u = \frac{\lambda + \log \lambda + 2}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$

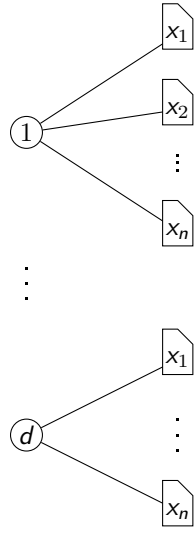
  (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)



Efficient Construction: Telescope (cont.)

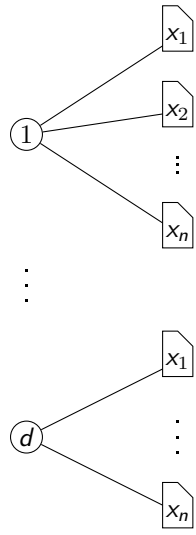
- Represent as d trees
- Prover runs DFS (instead of BFS)
- Union bound to analyze soundness:
  - soundness error

    $\text{“number”} \cdot \text{“probability”} = d \cdot n_{\mathrm{f}}^{u} \cdot \left(\frac{1}{n_{\mathrm{p}}}\right)^{u} \cdot q$
- Recursive formula to analyze completeness:
  - $f(0) = 1 - q$
  - $f(k+1) = \left((1 - \frac{1}{n_{\mathrm{p}}}) + \frac{1}{n_{\mathrm{p}}} \cdot f(k)\right)^{n_{\mathrm{p}}}$
  - completeness error $\left(f(u)\right)^{d}$
- Setting errors $\leq 2^{-\lambda}$, achieve proof size $u = \frac{\lambda + \log \lambda + 2}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)



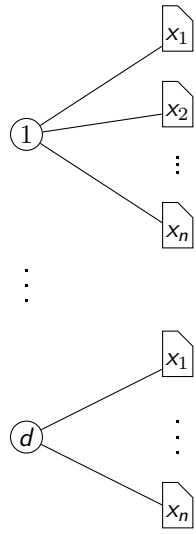
Efficient Construction: Telescope (cont.)

- To analyze prover running time:
  - u reachable vertices in a tree in expectation (by linearity of expectation)
  - a tree succeeds with probability $\Omega(1/\lambda)$
  - $\Longrightarrow$ expected running time $O(u \cdot \lambda \cdot n_{\mathrm{p}}) = O(\lambda^2 \cdot n_{\mathrm{p}})$
  - also $O(\lambda^3 \cdot n_{\mathrm{p}})$ w.h.p.



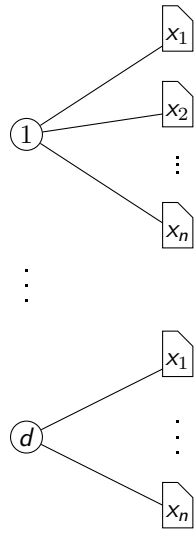
Efficient Construction: Telescope (cont.)

- To analyze prover running time:
  - u reachable vertices in a tree in expectation (by linearity of expectation)
  - a tree succeeds with probability $\Omega(1/\lambda)$
  - $\Longrightarrow$ expected running time $O(u \cdot \lambda \cdot n_{\mathrm{p}}) = O(\lambda^2 \cdot n_{\mathrm{p}})$
  - also $O(\lambda^3 \cdot n_{\mathrm{p}})$ w.h.p.



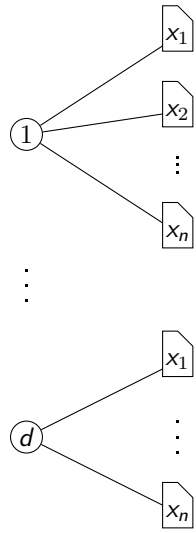
Efficient Construction: Telescope (cont.)

- To analyze prover running time:
  - u reachable vertices in a tree in expectation (by linearity of expectation)
  - a tree succeeds with probability $\Omega(1/\lambda)$
  - $\Longrightarrow$ expected running time $O(u \cdot \lambda \cdot n_{\mathrm{p}}) = O(\lambda^2 \cdot n_{\mathrm{p}})$
  - also $O(\lambda^3 \cdot n_{\mathrm{p}})$ w.h.p.



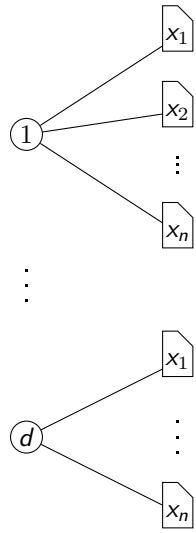
Efficient Construction: Telescope (cont.)

- To analyze prover running time:
  - u reachable vertices in a tree in expectation (by linearity of expectation)
  - a tree succeeds with probability $\Omega(1/\lambda)$
  - $\Longrightarrow$ expected running time $O(u \cdot \lambda \cdot n_{\mathrm{p}}) = O(\lambda^2 \cdot n_{\mathrm{p}})$
  - also $O(\lambda^3 \cdot n_{\mathrm{p}})$ w.h.p.



Efficient Construction: Telescope (cont.)

- To analyze prover running time:
  - u reachable vertices in a tree in expectation (by linearity of expectation)
  - a tree succeeds with probability $\Omega(1/\lambda)$
  - $\Longrightarrow$ expected running time $O(u \cdot \lambda \cdot n_{\mathrm{p}}) = O(\lambda^2 \cdot n_{\mathrm{p}})$
  - also $O(\lambda^3 \cdot n_{\mathrm{p}})$ w.h.p.

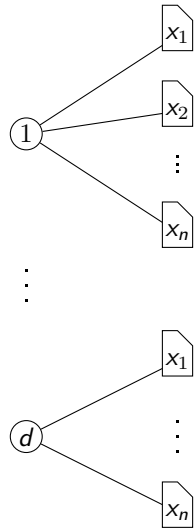


Coming next...

- ☑ Definitions
- Unweighted case
  - ☑ Prover-inefficient construction
  - ☑ Telescope construction
  - ☐ **Improving prover running time**
  - ☐ Decentralized setting
- ☐ Adding weights
- ☐ Applications

Improving Prover Runtime: Telescope with Prehashing

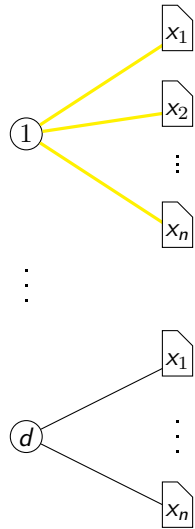
- Basic Telescope has running time $O(\lambda^2 \cdot n)$[1]
- Optimization inspired by balls-and-bins
- Hash $x_1, ..., x_n$ into n bins using $G_0(\cdot) \sim \mathrm{Unif}([n])$
- $(t, s_1, ..., s_j)$ has valid extensions in bin $G_1(t, s_1, ..., s_j) \sim \mathrm{Unif}([n])$



[1] $n = n_{\mathrm{p}}$

Improving Prover Runtime: Telescope with Prehashing

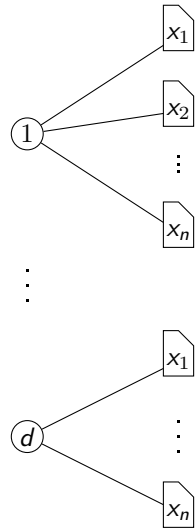
- Basic Telescope has running time $O(\lambda^2 \cdot n)$[1]
- Optimization inspired by balls-and-bins
- Hash $x_1, ..., x_n$ into n bins using $G_0(\cdot) \sim \mathrm{Unif}([n])$
- $(t, s_1, ..., s_j)$ has valid extensions in bin $G_1(t, s_1, ..., s_j) \sim \mathrm{Unif}([n])$



[1] $n = n_{\mathrm{p}}$

Improving Prover Runtime: Telescope with Prehashing

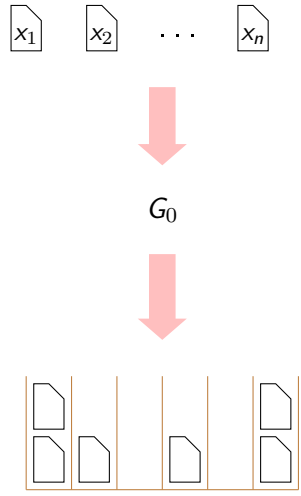
- Basic Telescope has running time $O(\lambda^2 \cdot n)$[1]
- Optimization inspired by balls-and-bins
- Hash $x_1, ..., x_n$ into n bins using $G_0(\cdot) \sim \mathrm{Unif}([n])$
- $(t, s_1, ..., s_j)$ has valid extensions in bin $G_1(t, s_1, ..., s_j) \sim \mathrm{Unif}([n])$



[1] $n = n_{\mathrm{p}}$

Improving Prover Runtime: Telescope with Prehashing

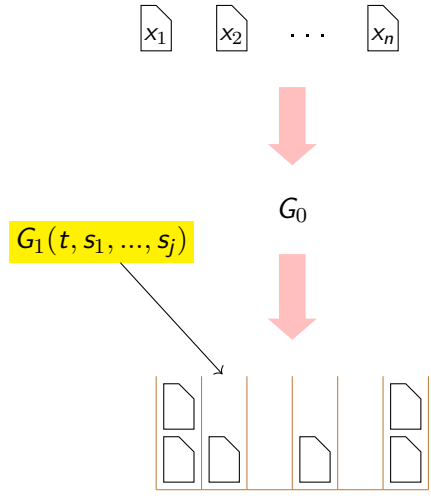
- Basic Telescope has running time $O(\lambda^2 \cdot n)$[1]
- Optimization inspired by balls-and-bins
- Hash $x_1, ..., x_n$ into n bins using $G_0(\cdot) \sim \mathrm{Unif}([n])$
- $(t, s_1, ..., s_j)$ has valid extensions in bin $G_1(t, s_1, ..., s_j) \sim \mathrm{Unif}([n])$



[1] $n = n_{\mathrm{p}}$

Improving Prover Runtime: Telescope with Prehashing

- Basic Telescope has running time $O(\lambda^2 \cdot n)$[1]
- Optimization inspired by balls-and-bins
- Hash $x_1, ..., x_n$ into n bins using $G_0(\cdot) \sim \mathrm{Unif}([n])$
- $(t, s_1, ..., s_j)$ has valid extensions in bin $G_1(t, s_1, ..., s_j) \sim \mathrm{Unif}([n])$



[1] $n = n_{\mathrm{p}}$

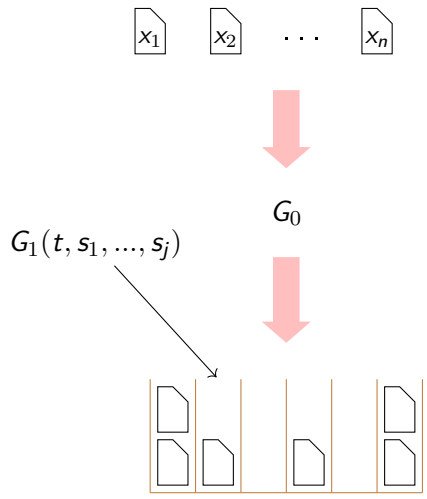
Improving Prover Runtime: Telescope with Prehashing

- Basic Telescope has running time $O(\lambda^2 \cdot n)$[1]
- Optimization inspired by balls-and-bins
- Hash $x_1, ..., x_n$ into n bins using $G_0(\cdot) \sim \mathrm{Unif}([n])$
- $(t, s_1, ..., s_j)$ has valid extensions in bin $G_1(\, t, s_1, ..., s_j\,) \sim \mathrm{Unif}([n])$

Valid proof: $(t, s_1, ..., s_u)$ s.t.

- $1 \leq t \leq d$;
- ~~$\forall 1 \leq i \leq u,\ H_1(t, s_1, ..., s_i) = 1$;~~
- $\forall 1 \leq i \leq u,\ G_1(t, s_1, ..., s_{i-1}) = G_0(s_i)$;
- $G_2(t, s_1, ..., s_u) = 1$;

$(G_0(\cdot), G_1(\cdot) \sim \mathrm{Unif}([n]),\ G_2(\cdot) \sim \mathrm{Bernoulli}(q))$



[1] $n = n_{\mathrm{p}}$

Improving Prover Runtime: Telescope with Prehashing (cont.)

- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_p \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_p + \lambda^2)$
  - Chernoff inequality approach (when $n_p \geq \Theta(\lambda^2)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_p + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_p \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_p/n_f)}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_p/n_f)}$)

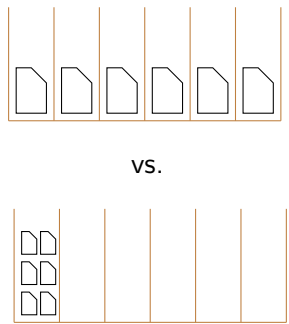
Valid proof: $(t, s_1, ..., s_u)$ s.t.

- $1 \leq t \leq d$;
- $\forall 1 \leq i \leq u$, $G_1(t, s_1, ..., s_{i-1}) = G_0(s_i)$;
- $G_2(t, s_1, ..., s_u) = 1$;

$(G_0(\cdot), G_1(\cdot) \sim \mathrm{Unif}([n]),$
$G_2(\cdot) \sim \mathrm{Bernoulli}(q))$

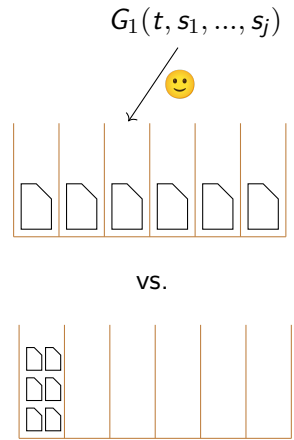
Improving Prover Runtime: Telescope with Prehashing (cont.)

- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_p \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_p + \lambda^2)$
  - Chernoff inequality approach (when $n_p \geq \Theta(\lambda^3)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_p + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_p \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_p/n_f)}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_p/n_f)}$)



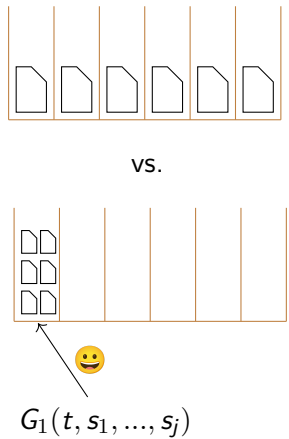
Improving Prover Runtime: Telescope with Prehashing (cont.)

- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_p \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_p + \lambda^2)$
  - Chernoff inequality approach (when $n_p \geq \Theta(\lambda^3)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_p + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_p \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_p/n_f)}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_p/n_f)}$)



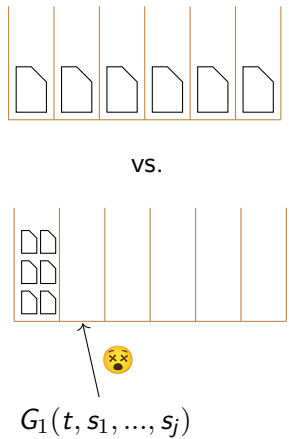
Improving Prover Runtime: Telescope with Prehashing (cont.)

- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_b \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_b + \lambda^2)$
  - Chernoff inequality approach (when $n_b \geq \Theta(\lambda^3)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_b + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_b \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_b/n_f)}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_b/n_f)}$)



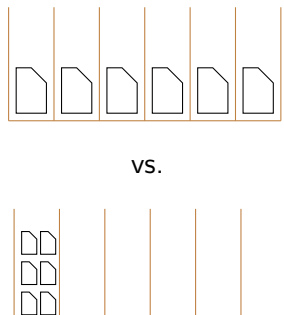
Improving Prover Runtime: Telescope with Prehashing (cont.)

- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_p \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_p + \lambda^2)$
  - Chernoff inequality approach (when $n_p \geq \Theta(\lambda^3)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_p + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_p \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_p/n_f)}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_p/n_f)}$)



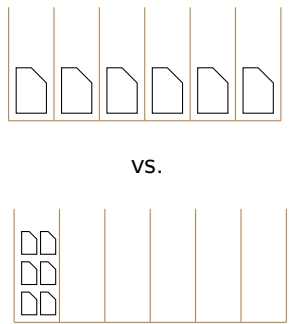
Improving Prover Runtime: Telescope with Prehashing (cont.)

- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_{\mathrm{p}} \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_{\mathrm{p}} + \lambda^2)$
  - Chernoff inequality approach (when $n_{\mathrm{p}} \geq \Theta(\lambda^3)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_{\mathrm{p}} + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_{\mathrm{p}} \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)



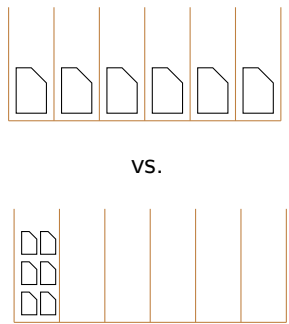
Improving Prover Runtime: Telescope with Prehashing (cont.)

- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_{\mathrm{p}} \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_{\mathrm{p}} + \lambda^2)$
  - Chernoff inequality approach (when $n_{\mathrm{p}} \geq \Theta(\lambda^3)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_{\mathrm{p}} + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_{\mathrm{p}} \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)



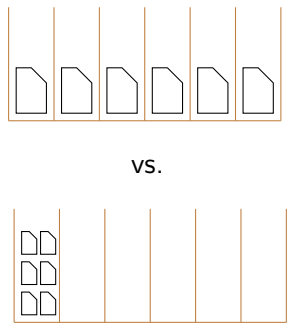
Improving Prover Runtime: Telescope with Prehashing (cont.)

- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_{\mathrm{p}} \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_{\mathrm{p}} + \lambda^2)$
  - Chernoff inequality approach (when $n_{\mathrm{p}} \geq \Theta(\lambda^3)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_{\mathrm{p}} + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_{\mathrm{p}} \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)



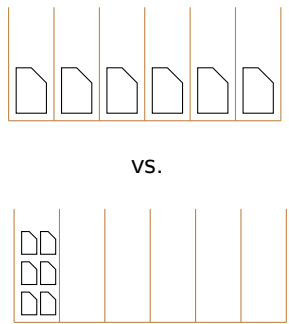
Improving Prover Runtime: Telescope with Prehashing (cont.)

- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_{\mathrm{p}} \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_{\mathrm{p}} + \lambda^2)$
  - Chernoff inequality approach (when $n_{\mathrm{p}} \geq \Theta(\lambda^3)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_{\mathrm{p}} + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_{\mathrm{p}} \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$
  (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)



Improving Prover Runtime: Telescope with Prehashing (cont.)

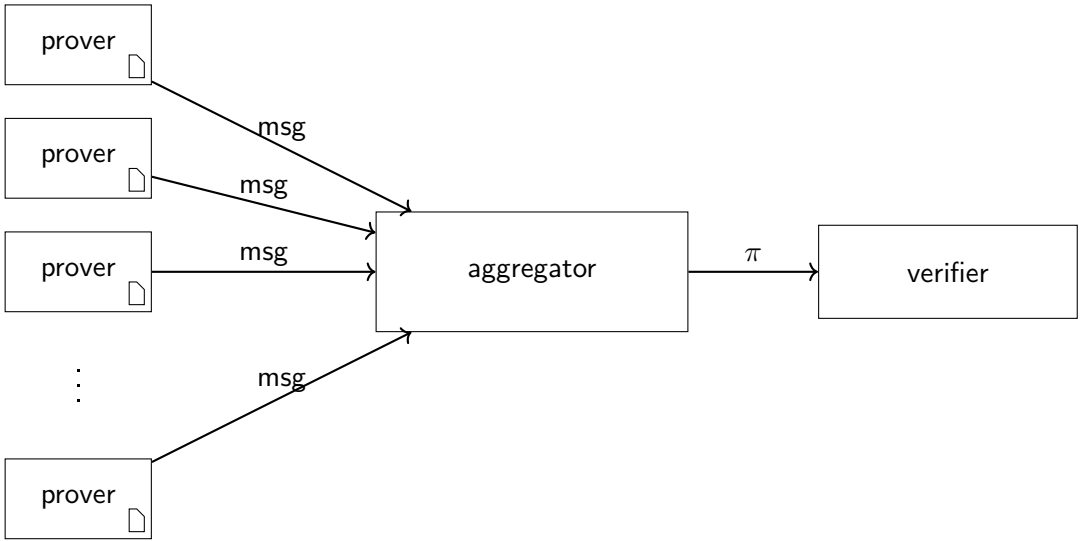
- Soundness analysis remains the same (union bound)
- Completeness analysis is tricky
- Need a "good" condition on "balls" distribution
- Use Poisson approximation + one of the following:
  - Markov's inequality approach (when $n_{\mathrm{p}} \leq \Theta(\lambda^2)$):
    - guarantees "good" condition with moderate prob.
    - get a scheme with completeness $1/2$; amplify
    - expected running time $O(n_{\mathrm{p}} + \lambda^2)$
  - Chernoff inequality approach (when $n_{\mathrm{p}} \geq \Theta(\lambda^3)$):
    - "good" condition w.h.p. by Chernoff-like analysis
    - running time $O(n_{\mathrm{p}} + Z)$ where $\mathbb{E}[Z] = \lambda^2$
  - Chernoff + amplify (when $\Theta(\lambda^2) \leq n_{\mathrm{p}} \leq \Theta(\lambda^3)$)
- Proof size $u = \frac{\lambda + \log \lambda + 4}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$ (lower bound: $u > \frac{\lambda - 3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$)



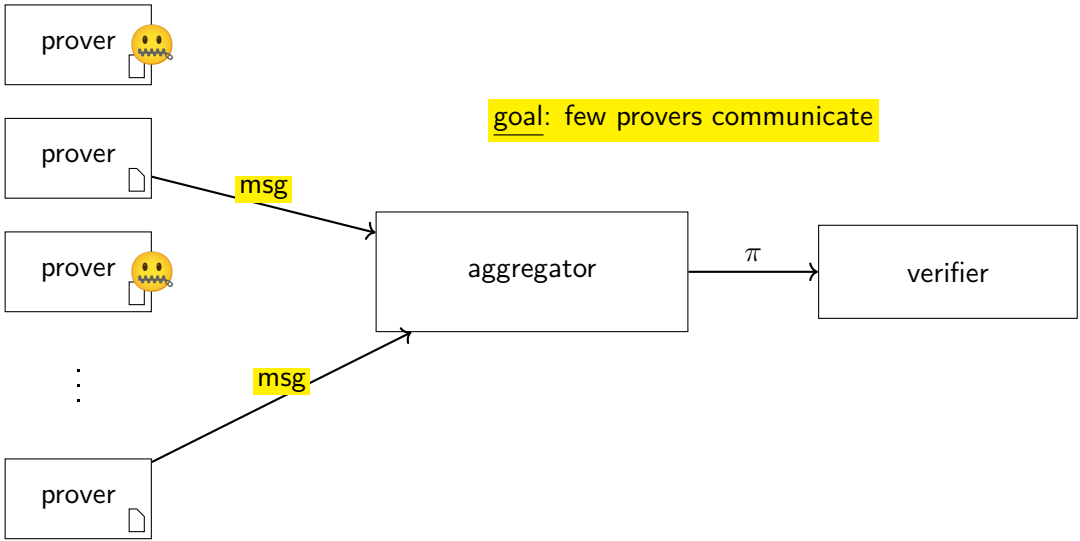
Coming next...

- ☑ Definitions
- Unweighted case
  - ☑ Prover-inefficient construction
  - ☑ Telescope construction
  - ☑ Improving prover running time
  - ☐ **Decentralized setting**
- ☐ Adding weights
- ☐ Applications

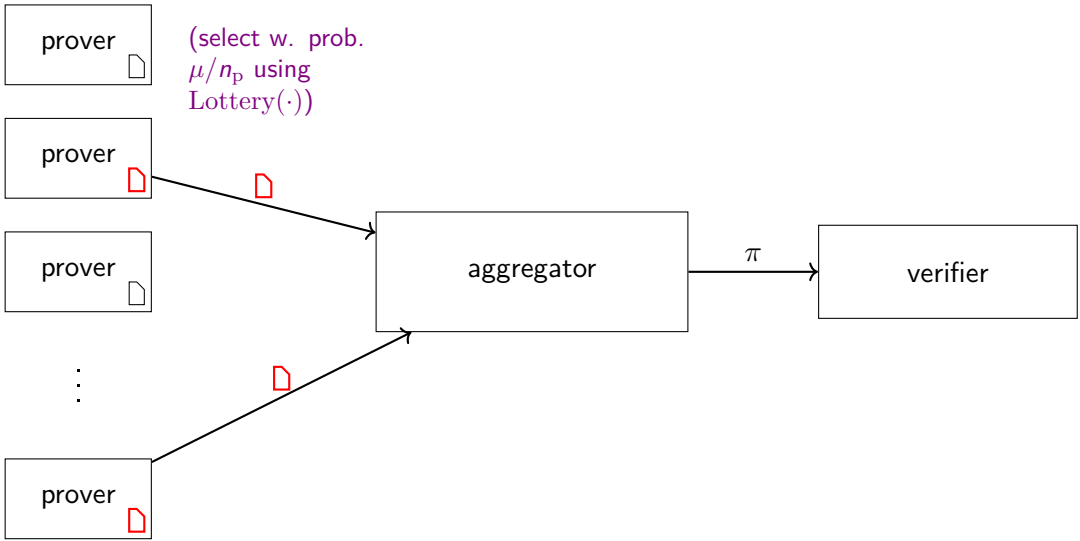
Decentralized ALBA



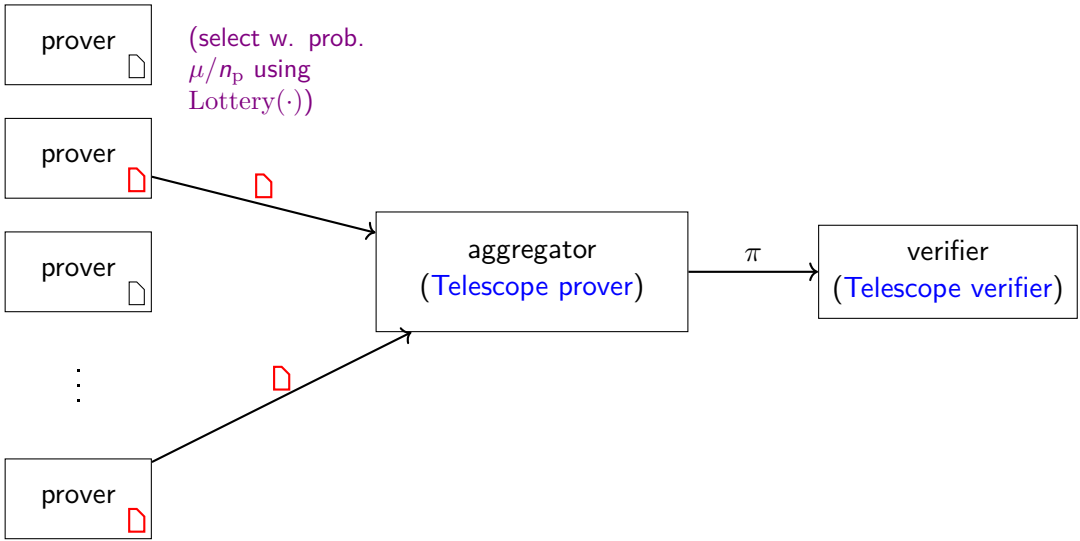
Decentralized ALBA



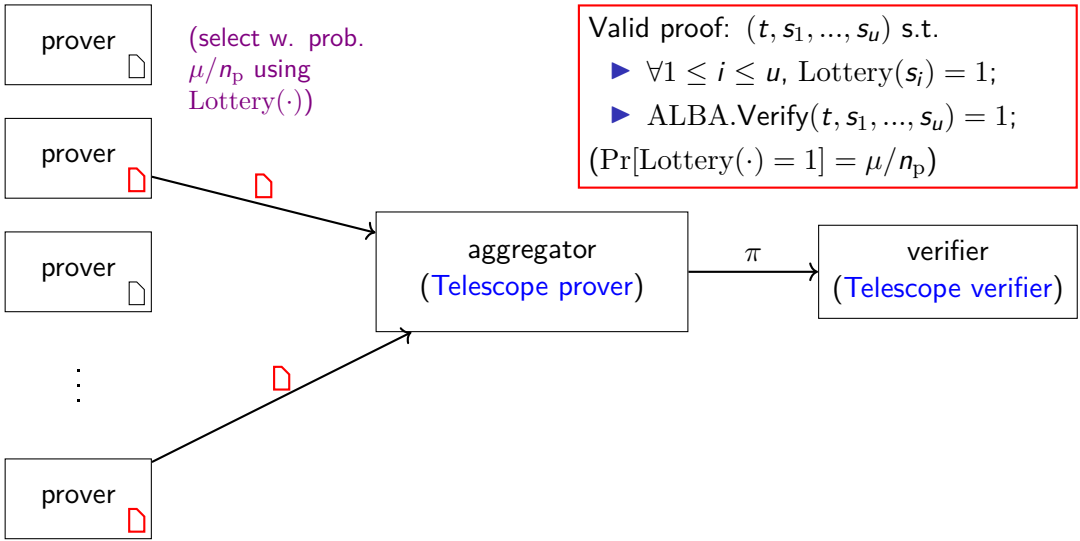
Decentralized ALBA



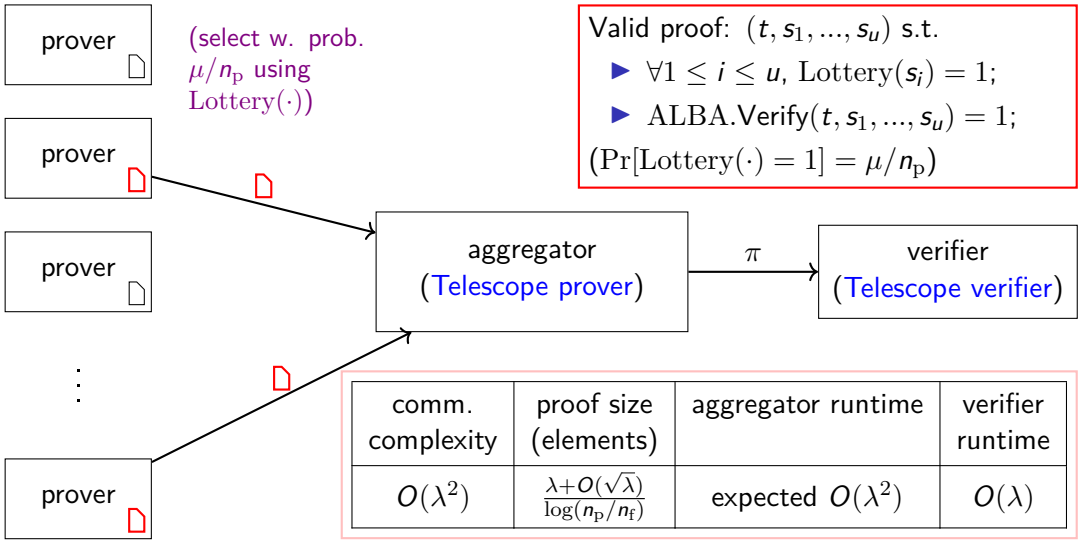
Decentralized ALBA



Decentralized ALBA



Decentralized ALBA



comm. complexity	proof size (elements)	aggregator runtime	verifier runtime
$O(\lambda^2)$	$\frac{\lambda + O(\sqrt{\lambda})}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$	expected $O(\lambda^2)$	$O(\lambda)$

Coming next...

- ☑ Definitions
- Unweighted case
  - ☑ Prover-inefficient construction
  - ☑ Telescope construction
  - ☑ Improving prover running time
  - ☑ Decentralized setting
- ☐ **Adding weights**
- ☐ Applications

Adding Weights

- Back to centralized setting
- Naive approach
  - turn weight-w element x into elements $(x, 1), ..., (x, w)$
  - inefficient (think of w as $\sim 2^{64}$)
- Lottery based approach

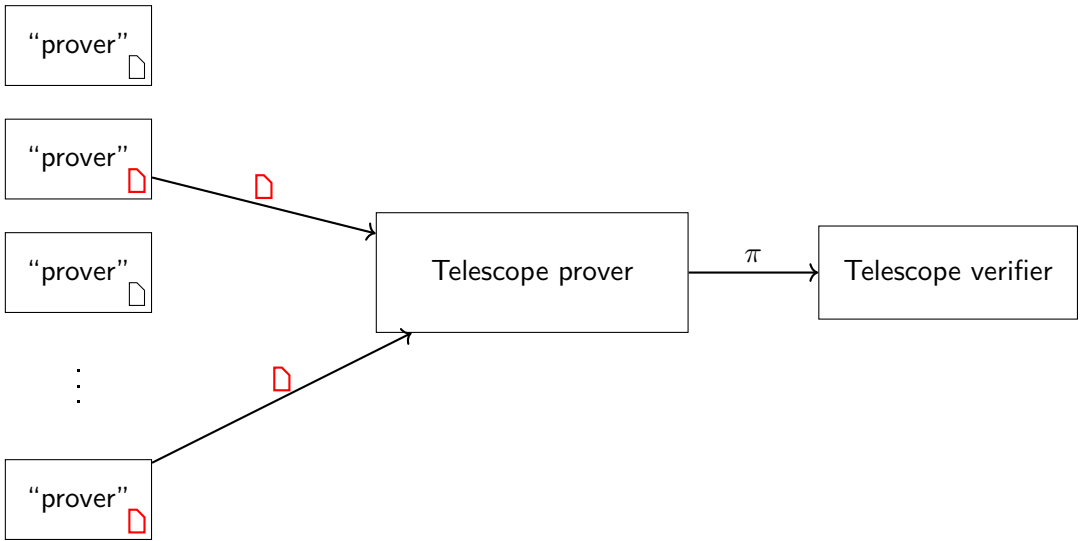
Adding Weights

- Back to centralized setting
- Naive approach
  - turn weight-w element x into elements $(x, 1), ..., (x, w)$
  - inefficient (think of w as $\sim 2^{64}$)
- Lottery based approach

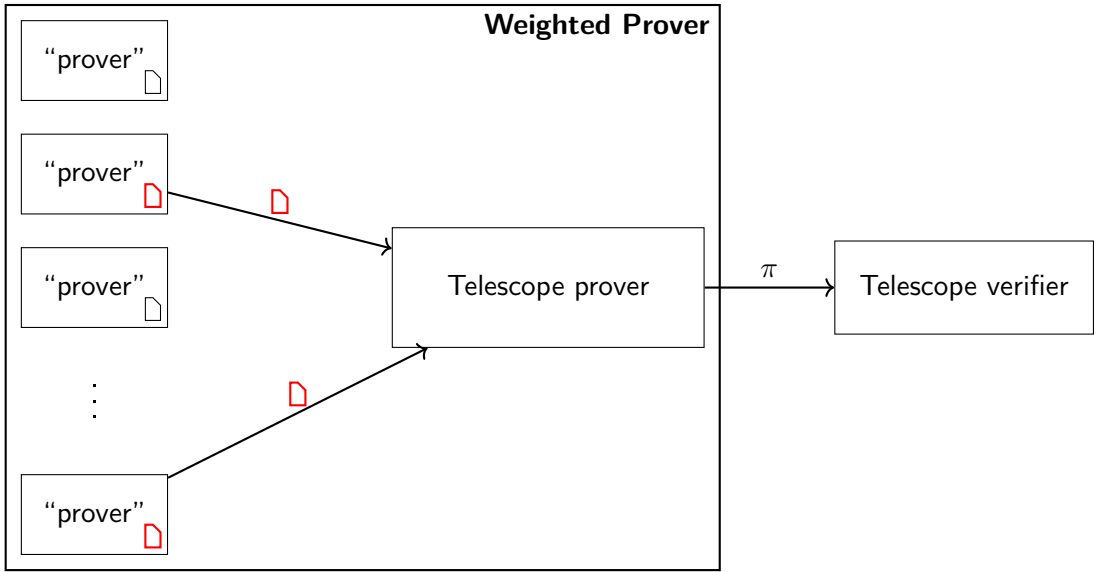
Adding Weights

- Back to centralized setting
- Naive approach
  - turn weight-w element x into elements $(x, 1), ..., (x, w)$
  - inefficient (think of w as $\sim 2^{64}$)
- Lottery based approach

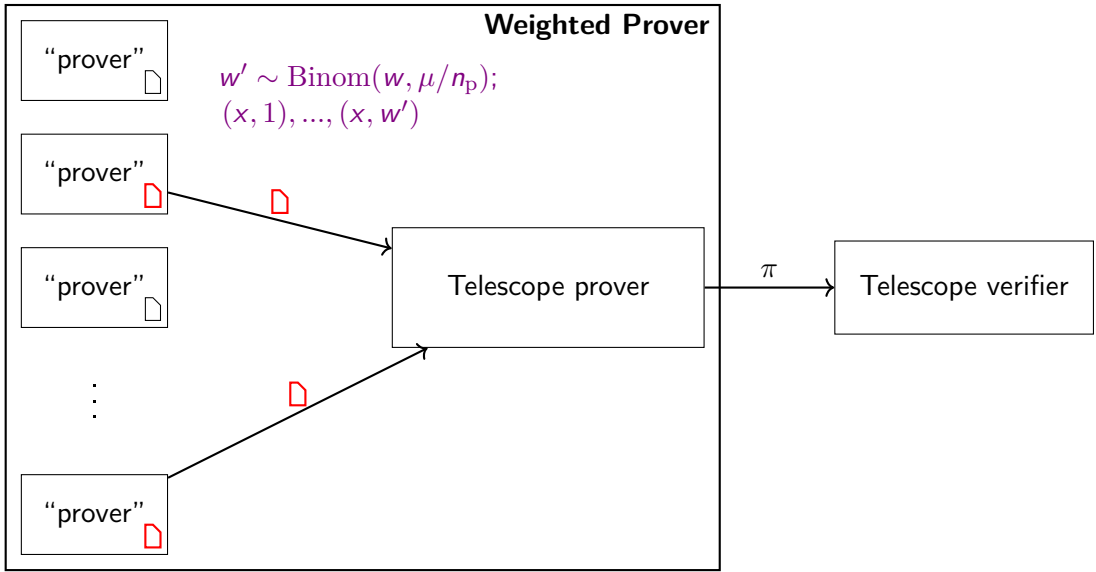
Adding Weights (cont.)



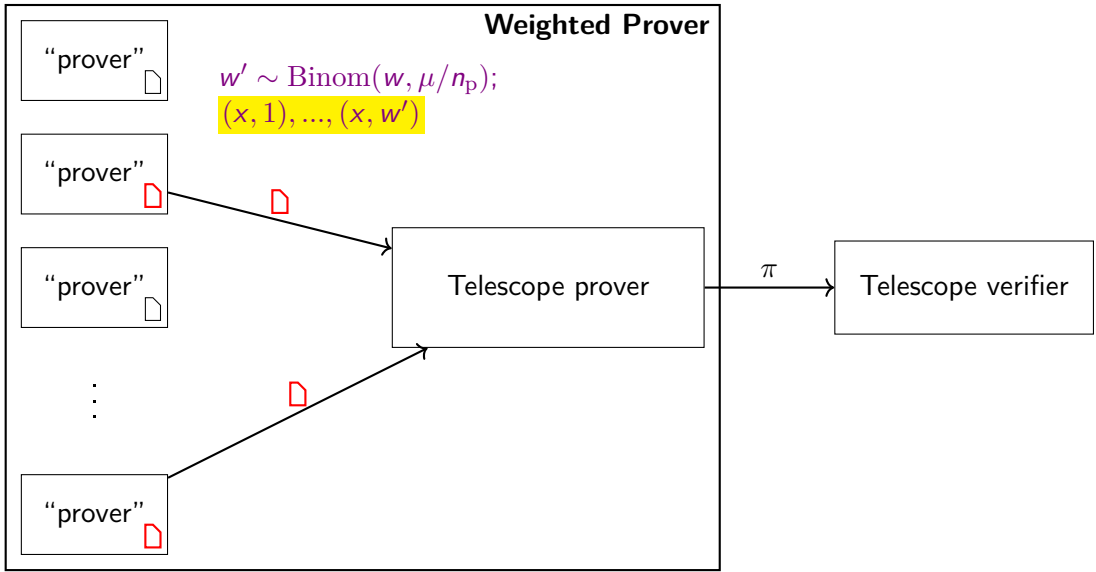
Adding Weights (cont.)



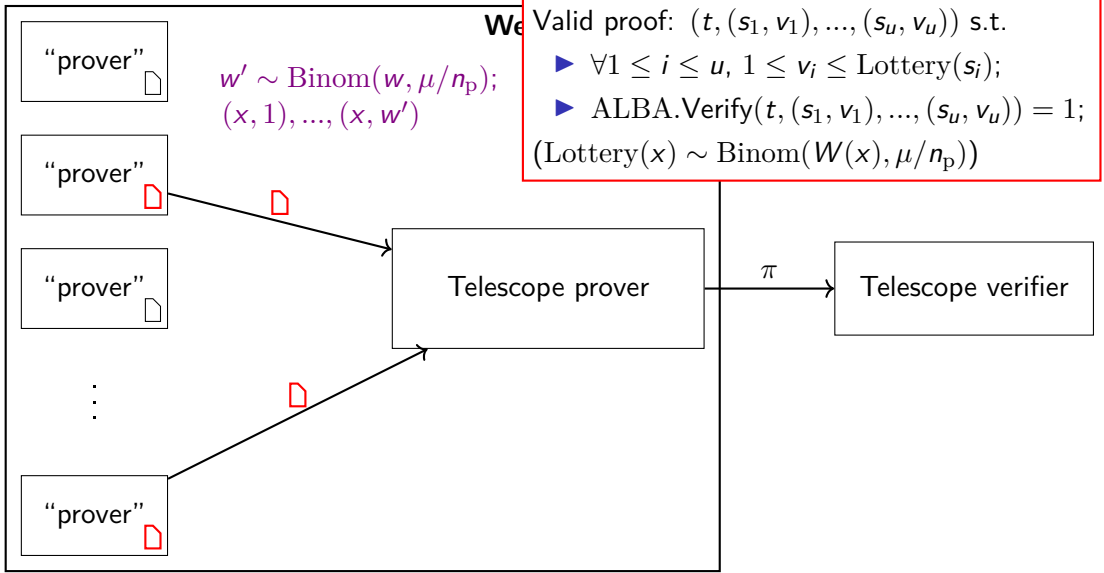
Adding Weights (cont.)



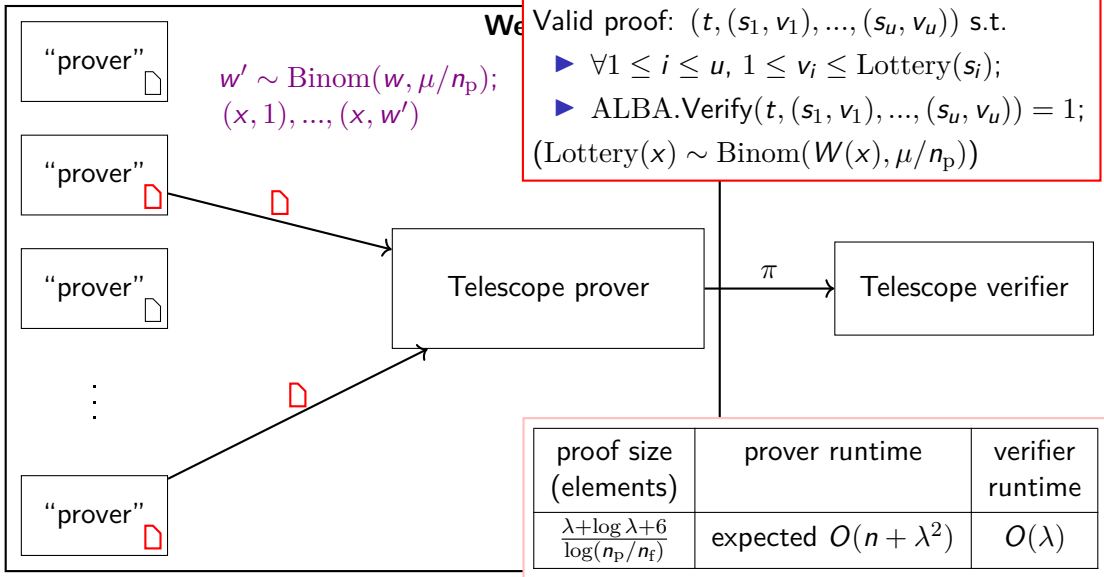
Adding Weights (cont.)



Adding Weights (cont.)



Adding Weights (cont.)



Valid proof: $(t,(s_1,v_1),...,(s_u,v_u))$ s.t.

- $\forall 1 \le i \le u,\ 1 \le v_i \le \mathrm{Lottery}(s_i)$;
- $\mathrm{ALBA.Verify}(t,(s_1,v_1),...,(s_u,v_u)) = 1$;

$(\mathrm{Lottery}(x) \sim \mathrm{Binom}(W(x), \mu/n_\mathrm{p}))$

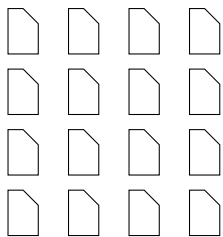
proof size (elements)	prover runtime	verifier runtime
$\frac{\lambda+\log\lambda+6}{\log(n_\mathrm{p}/n_\mathrm{f})}$	expected $O(n+\lambda^2)$	$O(\lambda)$

Coming next...

- ☑ Definitions
- Unweighted case
  - ☑ Prover-inefficient construction
  - ☑ Telescope construction
  - ☑ Improving prover running time
  - ☑ Decentralized setting
- ☑ Adding weights
- ☐ **Applications**

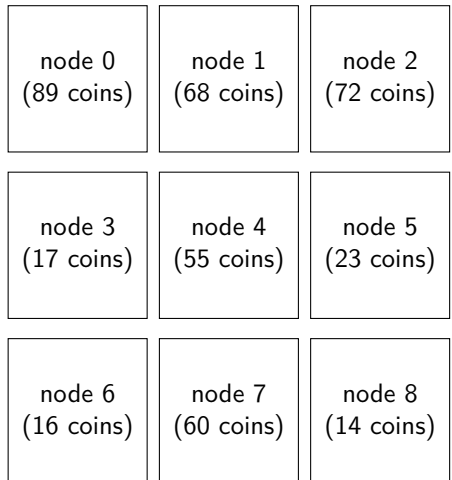
Application 1: Weighted Multisignatures

- Prove that sufficiently many parties signed a message
- Proof-of-stake blockchains: prove that parties with sufficient total stake (money) attested to blockchain state
  - assume 80% honest stake, 20% malicious stake $\Longrightarrow n_p/n_q = 4$
  - sign blockchain state
  - signature weight = node's stake
  - aggregate signatures using (decentralized) ALBA



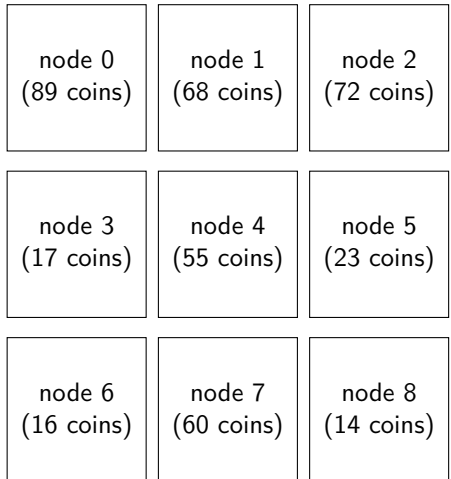
Application 1: Weighted Multisignatures

- Prove that sufficiently many parties signed a message
- Proof-of-stake blockchains: prove that parties with sufficient total stake (money) attested to blockchain state
  - assume 80% honest stake, 20% malicious stake $\Longrightarrow n_{\mathrm{p}}/n_{\mathrm{f}} = 4$
  - sign blockchain state
  - signature weight = node's stake
  - aggregate signatures using (decentralized) ALBA



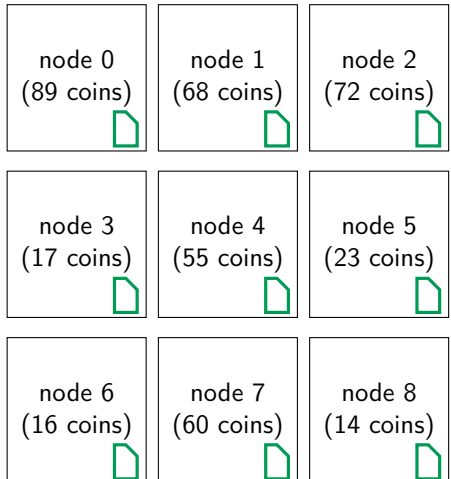
Application 1: Weighted Multisignatures

- Prove that sufficiently many parties signed a message
- Proof-of-stake blockchains: prove that parties with sufficient total stake (money) attested to blockchain state
  - assume 80% honest stake, 20% malicious stake $\implies n_{\mathrm{p}}/n_{\mathrm{f}} = 4$
  - sign blockchain state
  - signature weight = node's stake
  - aggregate signatures using (decentralized) ALBA



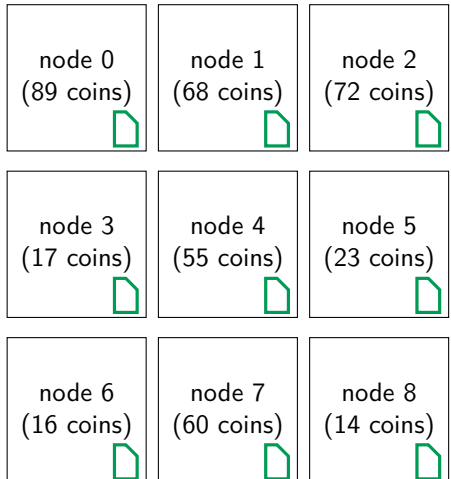
Application 1: Weighted Multisignatures

- Prove that sufficiently many parties signed a message
- Proof-of-stake blockchains: prove that parties with sufficient total stake (money) attested to blockchain state
  - assume 80% honest stake, 20% malicious stake $\implies n_{\mathrm{p}}/n_{\mathrm{f}} = 4$
  - sign blockchain state
  - signature weight = node's stake
  - aggregate signatures using (decentralized) ALBA



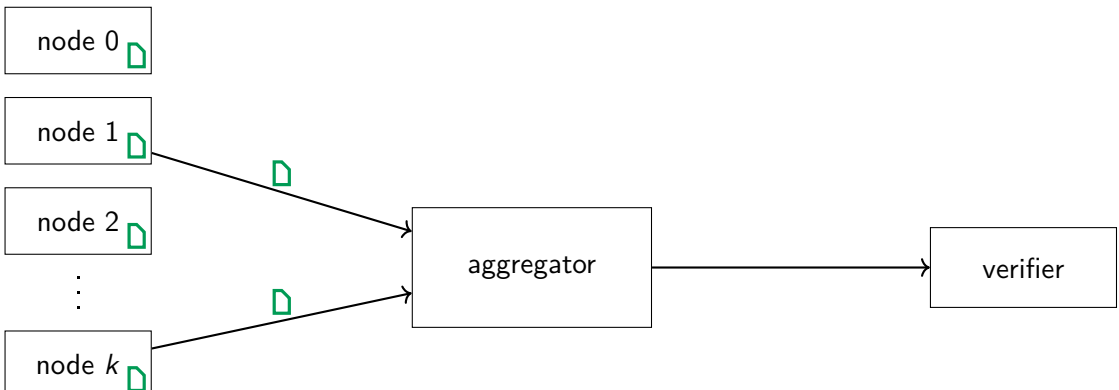
Application 1: Weighted Multisignatures

- Prove that sufficiently many parties signed a message
- Proof-of-stake blockchains: prove that parties with sufficient total stake (money) attested to blockchain state
  - assume 80% honest stake, 20% malicious stake $\implies n_{\mathrm{p}}/n_{\mathrm{f}} = 4$
  - sign blockchain state
  - signature weight = node's stake
  - aggregate signatures using (decentralized) ALBA



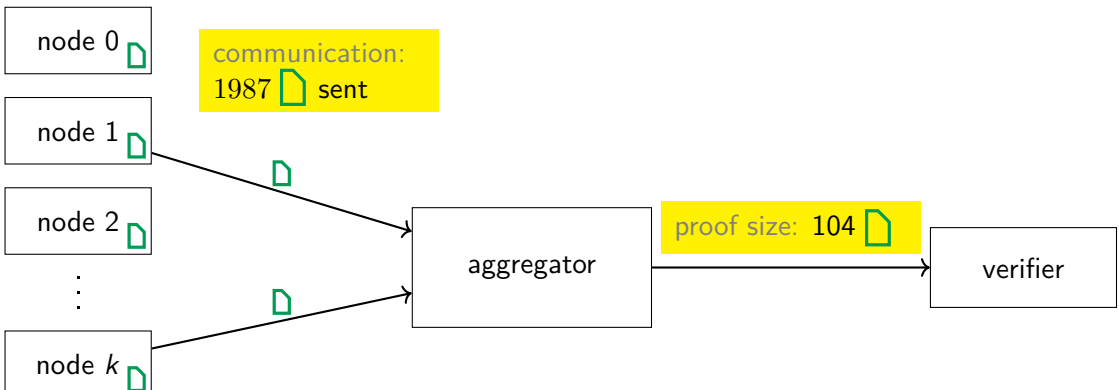
Application 1: Weighted Multisignatures (cont.)

small proof + small communication complexity



Application 1: Weighted Multisignatures (cont.)

small proof + small communication complexity, 128-bit security:



Application 2: Straight-Line Witness Extraction for SNARKs

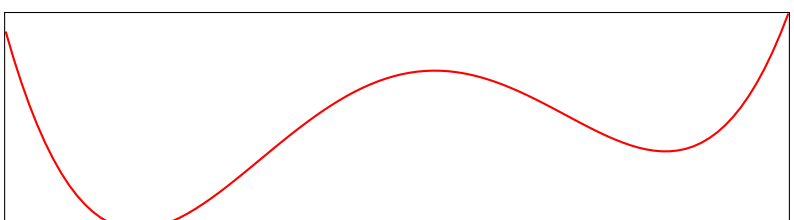
- [GKOPTT23] addresses Universal Composability (UC) for SNARKs
- UC requires witness-extraction without rewinding
- Since $|\pi| \ll |w|$, how to extract w without rewinding? Possible in RO model

Application 2: Straight-Line Witness Extraction for SNARKs

- [GKOPTT23] addresses Universal Composability (UC) for SNARKs
- UC requires witness-extraction without rewinding
- Since $|\pi| \ll |w|$, how to extract w without rewinding? Possible in RO model

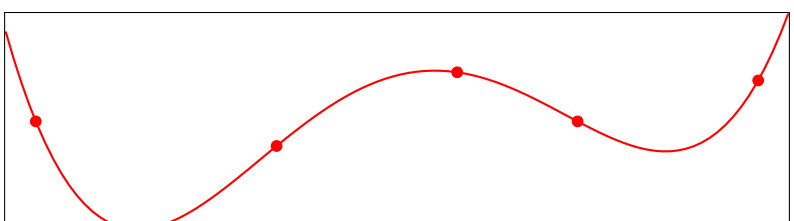
Application 2: Straight-Line Witness Extraction for SNARKs

- [GKOPTT23]:
  - Represent the witness as degree d polynomial
  - Commit using polynomial commitment scheme
  - Prove that RO was queried on $d+1$ valid points by querying λd points;
- Observation: An application of ALBA
- Result: prover queries $2d$ points instead (λ times faster prover); more modular construction



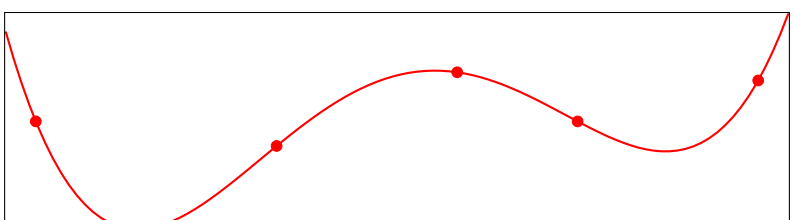
Application 2: Straight-Line Witness Extraction for SNARKs

- [GKOPTT23]:
  - Represent the witness as degree d polynomial
  - Commit using polynomial commitment scheme
  - Prove that RO was queried on $d+1$ valid points by querying λd points;
- Observation: An application of ALBA
- Result: prover queries $2d$ points instead (λ times faster prover); more modular construction



Application 2: Straight-Line Witness Extraction for SNARKs

- [GKOPTT23]:
  - Represent the witness as degree d polynomial
  - Commit using polynomial commitment scheme
  - Prove that RO was queried on $d+1$ valid points by querying λd points;
- Observation: An application of ALBA
- Result: prover queries $2d$ points instead (λ times faster prover); more modular construction



Outline

- ☑ Definitions
- Unweighted case
  - ☑ Prover-inefficient construction
  - ☑ Telescope construction
  - ☑ Improving prover running time
  - ☑ Decentralized setting
- ☑ Adding weights
- ☑ Applications

Additional slides:

- Lower bounds
- Table of all results
- Knowledge extraction (RO, straight-line)
- ALBA in CRS model

[GKOPTT23] Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. "Witness-Succinct Universally-Composable SNARKs". In: *EUROCRYPT 2023, Part II*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14005. LNCS. Springer, Heidelberg, Apr. 2023, pp. 315–346. DOI: 10.1007/978-3-031-30617-4_11.

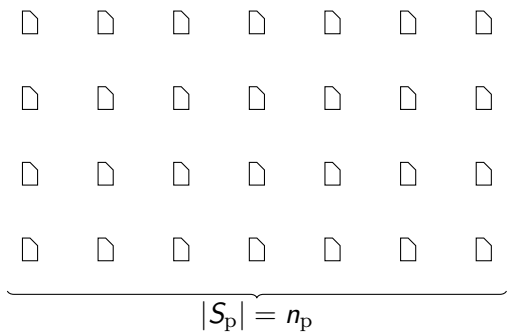
Lower Bounds

Theorem

Any ALBA scheme with completeness & soundness errors $\leq 2^{-\lambda}$ must have $\Omega(\lambda)$ proof size.

Proof outline

- (random variable) $\pi \leftarrow \mathrm{Prove}^H(S_\mathrm{p})$
- suppose $|\pi|$ is small
- take random $S_\mathrm{f} \subseteq S_\mathrm{p}$ with $|S_\mathrm{f}| = n_\mathrm{f}$
- $\Pr_{H,S_\mathrm{f}}[\pi \subseteq S_\mathrm{f}]$ is big
- by averaging argument $\exists S'_\mathrm{f}$ s.t. $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is big
- $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is small by soundness
- contradiction



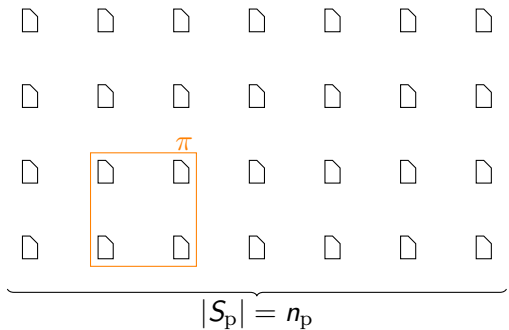
Lower Bounds

Theorem

Any ALBA scheme with completeness & soundness errors $\leq 2^{-\lambda}$ must have $\Omega(\lambda)$ proof size.

Proof outline

- (random variable) $\pi \leftarrow \mathsf{Prove}^H(S_\mathrm{p})$
- suppose $|\pi|$ is small
- take random $S_\mathrm{f} \subseteq S_\mathrm{p}$ with $|S_\mathrm{f}| = n_\mathrm{f}$
- $\Pr_{H,S_\mathrm{f}}[\pi \subseteq S_\mathrm{f}]$ is big
- by averaging argument $\exists S'_\mathrm{f}$ s.t. $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is big
- $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is small by soundness
- contradiction



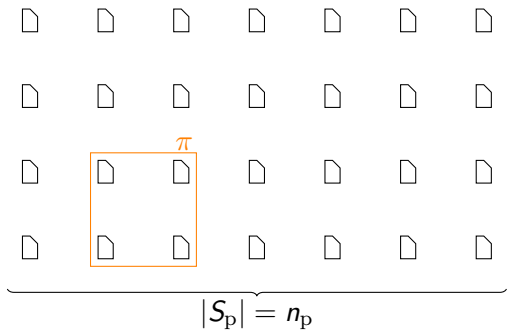
Lower Bounds

Theorem

Any ALBA scheme with completeness & soundness errors $\leq 2^{-\lambda}$ must have $\Omega(\lambda)$ proof size.

Proof outline

- (random variable) $\pi \leftarrow \mathsf{Prove}^H(S_\mathrm{p})$
- suppose $|\pi|$ is small
- take random $S_\mathrm{f} \subseteq S_\mathrm{p}$ with $|S_\mathrm{f}| = n_\mathrm{f}$
- $\Pr_{H,S_\mathrm{f}}[\pi \subseteq S_\mathrm{f}]$ is big
- by averaging argument $\exists S'_\mathrm{f}$ s.t. $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is big
- $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is small by soundness
- contradiction



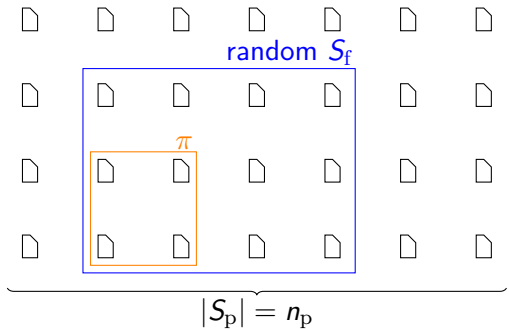
Lower Bounds

Theorem

Any ALBA scheme with completeness & soundness errors $\leq 2^{-\lambda}$ must have $\Omega(\lambda)$ proof size.

Proof outline

- (random variable) $\pi \leftarrow \mathsf{Prove}^H(S_\mathrm{p})$
- suppose $|\pi|$ is small
- take random $S_\mathrm{f} \subseteq S_\mathrm{p}$ with $|S_\mathrm{f}| = n_\mathrm{f}$
- $\Pr_{H,S_\mathrm{f}}[\pi \subseteq S_\mathrm{f}]$ is big
- by averaging argument $\exists S'_\mathrm{f}$ s.t. $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is big
- $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is small by soundness
- contradiction



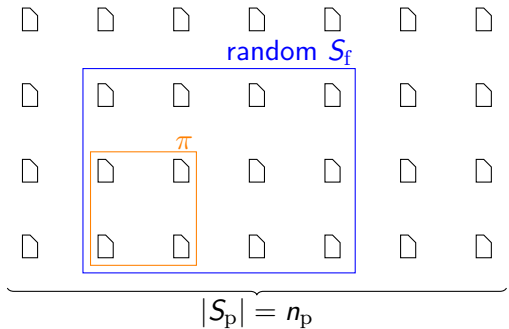
Lower Bounds

Theorem

Any ALBA scheme with completeness & soundness errors $\leq 2^{-\lambda}$ must have $\Omega(\lambda)$ proof size.

Proof outline

- (random variable) $\pi \leftarrow \mathsf{Prove}^H(S_\mathrm{p})$
- suppose $|\pi|$ is small
- take random $S_\mathrm{f} \subseteq S_\mathrm{p}$ with $|S_\mathrm{f}| = n_\mathrm{f}$
- $\Pr_{H,S_\mathrm{f}}[\pi \subseteq S_\mathrm{f}]$ is big
- by averaging argument $\exists S'_\mathrm{f}$ s.t. $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is big
- $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is small by soundness
- contradiction



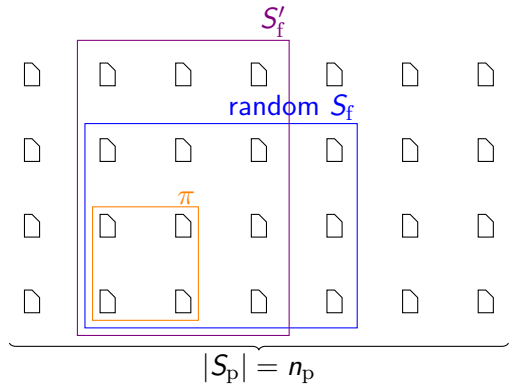
Lower Bounds

Theorem

Any ALBA scheme with completeness & soundness errors $\leq 2^{-\lambda}$ must have $\Omega(\lambda)$ proof size.

Proof outline

- (random variable) $\pi \leftarrow \mathsf{Prove}^H(S_\mathrm{p})$
- suppose $|\pi|$ is small
- take random $S_\mathrm{f} \subseteq S_\mathrm{p}$ with $|S_\mathrm{f}| = n_\mathrm{f}$
- $\Pr_{H,S_\mathrm{f}}[\pi \subseteq S_\mathrm{f}]$ is big
- by averaging argument $\exists S'_\mathrm{f}$ s.t. $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is big
- $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is small by soundness
- contradiction



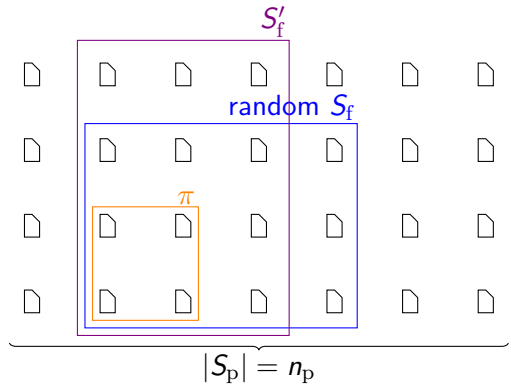
Lower Bounds

Theorem

Any ALBA scheme with completeness & soundness errors $\leq 2^{-\lambda}$ must have $\Omega(\lambda)$ proof size.

Proof outline

- (random variable) $\pi \leftarrow \mathsf{Prove}^H(S_\mathrm{p})$
- suppose $|\pi|$ is small
- take random $S_\mathrm{f} \subseteq S_\mathrm{p}$ with $|S_\mathrm{f}| = n_\mathrm{f}$
- $\Pr_{H,S_\mathrm{f}}[\pi \subseteq S_\mathrm{f}]$ is big
- by averaging argument $\exists S'_\mathrm{f}$ s.t. $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is big
- $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is small by soundness
- contradiction



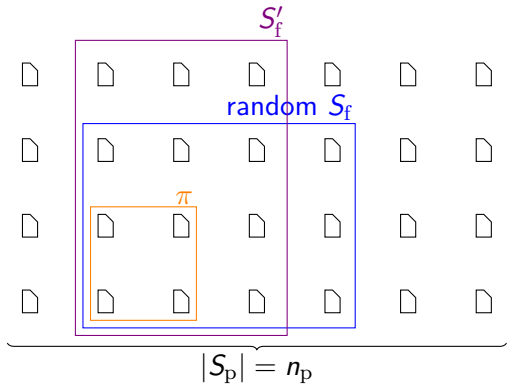
Lower Bounds

Theorem

Any ALBA scheme with completeness & soundness errors $\leq 2^{-\lambda}$ must have $\Omega(\lambda)$ proof size.

Proof outline

- (random variable) $\pi \leftarrow \mathsf{Prove}^H(S_\mathrm{p})$
- suppose $|\pi|$ is small
- take random $S_\mathrm{f} \subseteq S_\mathrm{p}$ with $|S_\mathrm{f}| = n_\mathrm{f}$
- $\Pr_{H,S_\mathrm{f}}[\pi \subseteq S_\mathrm{f}]$ is big
- by averaging argument $\exists S'_\mathrm{f}$ s.t. $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is big
- $\Pr_H[\pi \subseteq S'_\mathrm{f}]$ is small by soundness
- contradiction



Lower Bounds (cont.)

scheme	proof size (elements)	comm. rounds	comm. complexity
ALBA	$> \frac{\lambda-3}{\log(n_\mathrm{p}/n_\mathrm{f})}$		
decentralized ALBA[1]	$> \frac{\lambda+\Omega(\sqrt{\lambda})}{\log(n_\mathrm{p}/n_\mathrm{f})}$	1	$O(\lambda^2)$

[1] in a simplified model

Lower Bounds (cont.)

scheme	proof size (elements)	comm. rounds	comm. complexity
ALBA	$> \frac{\lambda-3}{\log(n_\mathrm{p}/n_\mathrm{f})}$		
decentralized ALBA[1]	$> \frac{\lambda+\Omega(\sqrt{\lambda})}{\log(n_\mathrm{p}/n_\mathrm{f})}$	1	$O(\lambda^2)$

[1] in a simplified model

All Results

setting	lower bound	Telescope
ALBA	$> \frac{\lambda-3}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$	$\frac{\lambda+\log\lambda+6}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$
decentralized ALBA; communication $O(\lambda^2)$	$> \frac{\lambda+\Omega(\sqrt{\lambda})}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$	$\frac{\lambda+O(\sqrt{\lambda})}{\log(n_{\mathrm{p}}/n_{\mathrm{f}})}$

Table: Proof size

scheme	average	worst case
Telescope prover	$O(n+\lambda^2)$	$O(n+\lambda^3)$
decentralized Telescope aggregator; communication $O(\lambda^2)$	$O(\lambda^2)$	$O(\lambda^3)$
Telescope verifier	$O(\lambda)$	$O(\lambda)$

Table: Running time

Knowledge Extraction (RO, straight-line)

- $\mathcal{A}^{H,W}$ produces valid proof with probability $2^{-\lambda} + \varepsilon$
- **goal**: extract $> n_{\mathrm{f}}$ weight-1 elements with probability ε

Theorem
$\mathrm{Extract}^{H,W,\mathcal{A}}$ *extracts* $> n_f$ *elements with probability* ε

Proof.
$2^{-\lambda} + \varepsilon =$
$\Pr[\mathcal{A} \text{ succeeds}] \leq$
$\Pr[\mathcal{A} \text{ queries} \leq n_{\mathrm{f}} \text{ elements and succeeds}] +$
$\Pr[\mathcal{A} \text{ queries} > n_{\mathrm{f}} \text{ elements}]$ □

Algorithm $\mathrm{Extract}^{H,W,\mathcal{A}}$

run $\pi \leftarrow \mathcal{A}^{H,W}()$ and $\mathrm{Verify}^{H,W}(\pi)$ and observe their RO transcript τ;
$S_{\mathrm{f}} := \emptyset$;
for x queried to H_1 or H_2 in τ **do**
  if $W(x) = 1$ **then**
    add x to S_{f};
return S_{f};

Knowledge Extraction (RO, straight-line)

- $\mathcal{A}^{H,W}$ produces valid proof with probability $2^{-\lambda} + \varepsilon$
- **goal**: extract $> n_{\mathrm{f}}$ weight-1 elements with probability ε

Theorem
$\mathrm{Extract}^{H,W,\mathcal{A}}$ *extracts* $> n_f$ *elements with probability* ε

Proof.
$2^{-\lambda} + \varepsilon =$
$\Pr[\mathcal{A} \text{ succeeds}] \leq$
$\Pr[\mathcal{A} \text{ queries} \leq n_{\mathrm{f}} \text{ elements and succeeds}] +$
$\Pr[\mathcal{A} \text{ queries} > n_{\mathrm{f}} \text{ elements}]$ □

Algorithm $\mathrm{Extract}^{H,W,\mathcal{A}}$

run $\pi \leftarrow \mathcal{A}^{H,W}()$ and $\mathrm{Verify}^{H,W}(\pi)$ and observe their RO transcript τ;
$S_{\mathrm{f}} := \emptyset$;
for x queried to H_1 or H_2 in τ **do**
    if $W(x) = 1$ **then**
        add x to S_{f};
return S_{f};

Knowledge Extraction (RO, straight-line)

- $\mathcal{A}^{H,W}$ produces valid proof with probability $2^{-\lambda} + \varepsilon$
- **goal**: extract $> n_{\mathrm{f}}$ weight-1 elements with probability ε

Theorem
$\mathrm{Extract}^{H,W,\mathcal{A}}$ *extracts* $> n_f$ *elements with probability* ε

Proof.
$2^{-\lambda} + \varepsilon =$
$\Pr[\mathcal{A} \text{ succeeds}] \leq$
$\Pr[\mathcal{A} \text{ queries} \leq n_{\mathrm{f}} \text{ elements and succeeds}] +$
$\Pr[\mathcal{A} \text{ queries} > n_{\mathrm{f}} \text{ elements}]$ □

Algorithm $\mathrm{Extract}^{H,W,\mathcal{A}}$

run $\pi \leftarrow \mathcal{A}^{H,W}()$ and $\mathrm{Verify}^{H,W}(\pi)$ and observe their RO transcript τ;
$S_{\mathrm{f}} := \emptyset$;
for x queried to H_1 or H_2 in τ **do**
    if $W(x) = 1$ **then**
        add x to S_{f};
return S_{f};

Knowledge Extraction (RO, straight-line)

- $\mathcal{A}^{H,W}$ produces valid proof with probability $2^{-\lambda} + \varepsilon$
- **goal**: extract $> n_{\mathrm{f}}$ weight-1 elements with probability ε

Theorem
$\mathrm{Extract}^{H,W,\mathcal{A}}$ *extracts* $> n_f$ *elements with probability* ε

Proof.
$2^{-\lambda} + \varepsilon =$
$\Pr[\mathcal{A} \text{ succeeds}] \leq$
$\Pr[\mathcal{A} \text{ queries} \leq n_{\mathrm{f}} \text{ elements and succeeds}] +$
$\Pr[\mathcal{A} \text{ queries} > n_{\mathrm{f}} \text{ elements}]$ □

Algorithm $\mathrm{Extract}^{H,W,\mathcal{A}}$

run $\pi \leftarrow \mathcal{A}^{H,W}()$ and $\mathrm{Verify}^{H,W}(\pi)$ and observe their RO transcript τ;
$S_{\mathrm{f}} := \emptyset$;
for x queried to H_1 or H_2 in τ **do**
    if $W(x) = 1$ **then**
        add x to S_{f};
return S_{f};

Knowledge Extraction (RO, straight-line)

- $\mathcal{A}^{H,W}$ produces valid proof with probability $2^{-\lambda} + \varepsilon$
- **goal**: extract $> n_{\mathrm{f}}$ weight-1 elements with probability ε

Theorem
$\mathrm{Extract}^{H,W,\mathcal{A}}$ *extracts* $> n_f$ *elements with probability* ε

Proof.
$2^{-\lambda} + \varepsilon =$
$\Pr[\mathcal{A} \text{ succeeds}] \leq$
$\Pr[\mathcal{A} \text{ queries} \leq n_{\mathrm{f}} \text{ elements and succeeds}] +$
$\Pr[\mathcal{A} \text{ queries} > n_{\mathrm{f}} \text{ elements}]$ □

Algorithm $\mathrm{Extract}^{H,W,\mathcal{A}}$

run $\pi \leftarrow \mathcal{A}^{H,W}()$ and $\mathrm{Verify}^{H,W}(\pi)$ and observe their RO transcript τ;
$S_{\mathrm{f}} := \emptyset$;
for x queried to H_1 or H_2 in τ **do**
    if $W(x) = 1$ **then**
        add x to S_{f};
return S_{f};

Knowledge Extraction (RO, straight-line)

- $\mathcal{A}^{H,W}$ produces valid proof with probability $2^{-\lambda} + \varepsilon$
- **goal**: extract $> n_{\mathrm{f}}$ weight-1 elements with probability ε

Theorem
$\mathsf{Extract}^{H,W,\mathcal{A}}$ *extracts* $> n_f$ *elements with probability* ε

Proof.
$2^{-\lambda} + \varepsilon =$
$\Pr[\mathcal{A} \text{ succeeds}] \leq$
$\Pr[\mathcal{A} \text{ queries} \leq n_{\mathrm{f}} \text{ elements and succeeds}] +$
$\Pr[\mathcal{A} \text{ queries} > n_{\mathrm{f}} \text{ elements}]$ □

Algorithm $\mathsf{Extract}^{H,W,\mathcal{A}}$

run $\pi \leftarrow \mathcal{A}^{H,W}()$ and $\mathsf{Verify}^{H,W}(\pi)$ and observe their RO transcript τ;
$S_{\mathrm{f}} := \emptyset$;
for x queried to H_1 or H_2 in τ **do**
    if $W(x) = 1$ **then**
        add x to S_{f};
return S_{f};

Knowledge Extraction (RO, straight-line)

- $\mathcal{A}^{H,W}$ produces valid proof with probability $2^{-\lambda} + \varepsilon$
- **goal**: extract $> n_{\mathrm{f}}$ weight-1 elements with probability ε

Theorem
$\mathsf{Extract}^{H,W,\mathcal{A}}$ *extracts* $> n_f$ *elements with probability* ε

Proof.
$2^{-\lambda} + \varepsilon =$
$\Pr[\mathcal{A} \text{ succeeds}] \leq$
$\Pr[\mathcal{A} \text{ queries} \leq n_{\mathrm{f}} \text{ elements and succeeds}] +$
$\Pr[\mathcal{A} \text{ queries} > n_{\mathrm{f}} \text{ elements}]$ □

Algorithm $\mathsf{Extract}^{H,W,\mathcal{A}}$

run $\pi \leftarrow \mathcal{A}^{H,W}()$ and $\mathsf{Verify}^{H,W}(\pi)$ and observe their RO transcript τ;
$S_{\mathrm{f}} := \emptyset$;
for x queried to H_1 or H_2 in τ **do**
  if $W(x) = 1$ **then**
    add x to S_{f};
return S_{f};

Knowledge Extraction (RO, straight-line) (cont.)

Lemma

$\Pr[\mathcal{A}\ \textit{queries} \leq n_f\ \textit{elements and succeeds}] \leq 2^{-\lambda}$

Issue: adaptive adversary $\Longrightarrow$ union bound doesn't work

Proof.

- think about indices as inputs to H_1, H_2
- union bound applies

□

We achieved:

- information-theoretic security
  when weight function is fixed in advance
- can get security assuming a bound on the number of RO queries
  when weight function can depend on RO

Knowledge Extraction (RO, straight-line) (cont.)

Lemma

$\Pr[\mathcal{A}\ \textit{queries} \leq n_f\ \textit{elements and succeeds}] \leq 2^{-\lambda}$

Issue: adaptive adversary $\Longrightarrow$ union bound doesn't work

Proof.

- think about indices as inputs to H_1, H_2
- union bound applies

□

We achieved:

- information-theoretic security
  when weight function is fixed in advance
- can get security assuming a bound on the number of RO queries
  when weight function can depend on RO

Knowledge Extraction (RO, straight-line) (cont.)

Lemma

$\Pr[\mathcal{A}\ \textit{queries} \leq n_f\ \textit{elements and succeeds}] \leq 2^{-\lambda}$

Issue: adaptive adversary $\Longrightarrow$ union bound doesn't work

Proof.

- think about indices as inputs to H_1, H_2
- union bound applies

□

We achieved:

- information-theoretic security
  when weight function is fixed in advance
- can get security assuming a bound on the number of RO queries
  when weight function can depend on RO

Telescope in CRS[3] model

- Replace RO with PRF[2]
- CRS = PRF key
- PRF key need not be secret

[2] Pseudorandom Function
[3] Common Reference String

Knowledge Extraction in CRS model

- Knowledge extraction via rewinding:
  - suppose $\mathcal{A}^W(\mathrm{crs})$ succeeds with probability $2^{-\lambda} + \varepsilon \Longrightarrow$ extract with probability ε
  - while $|S_\mathrm{f}| \leq n_\mathrm{f}$, $\mathcal{A}^W(\mathrm{crs})$ outputs weight-1 elements outside of S_f with probability ε
- We achieve:
  - security when weight function is fixed in advance
  - can get security when weight function is not fixed in advance if PRF key is chosen by RO

Algorithm $\mathrm{Extract}^W(\mathcal{A})$

$S_\mathrm{f} := \emptyset$;
while $|S_\mathrm{f}| \leq n_\mathrm{f}$ **do**
  $\mathrm{crs} \leftarrow \mathrm{GenCRS}()$;
  $\pi \leftarrow \mathcal{A}^W(\mathrm{crs})$;
    ▷ $\Pr[\pi \cap W \setminus S_f \geq 1] \geq \varepsilon$
  $S_\mathrm{f} := S_\mathrm{f} \cup (\pi \cap W)$;
return S_f;

Knowledge Extraction in CRS model

- Knowledge extraction via rewinding:
  - suppose $\mathcal{A}^W(\mathrm{crs})$ succeeds with probability $2^{-\lambda} + \varepsilon \Longrightarrow$ extract with probability ε
  - while $|S_\mathrm{f}| \leq n_\mathrm{f}$, $\mathcal{A}^W(\mathrm{crs})$ outputs weight-1 elements outside of S_f with probability ε
- We achieve:
  - security when weight function is fixed in advance
  - can get security when weight function is not fixed in advance if PRF key is chosen by RO

Algorithm $\mathsf{Extract}^W(\mathcal{A})$

$S_\mathrm{f} \coloneqq \emptyset$;
while $|S_\mathrm{f}| \leq n_\mathrm{f}$ **do**
  $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$;
  $\pi \leftarrow \mathcal{A}^W(\mathrm{crs})$;
    ▷ $\Pr[\pi \cap W \setminus S_f \geq 1] \geq \varepsilon$
  $S_\mathrm{f} \coloneqq S_\mathrm{f} \cup (\pi \cap W)$;
return S_f;

Knowledge Extraction in CRS model

- Knowledge extraction via rewinding:
  - suppose $\mathcal{A}^W(\mathrm{crs})$ succeeds with probability $2^{-\lambda} + \varepsilon \Longrightarrow$ extract with probability ε
  - while $|S_\mathrm{f}| \leq n_\mathrm{f}$, $\mathcal{A}^W(\mathrm{crs})$ outputs weight-1 elements outside of S_f with probability ε
- We achieve:
  - security when weight function is fixed in advance
  - can get security when weight function is not fixed in advance if PRF key is chosen by RO

Algorithm $\mathsf{Extract}^W(\mathcal{A})$

$S_\mathrm{f} \coloneqq \emptyset$;
while $|S_\mathrm{f}| \leq n_\mathrm{f}$ **do**
  $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$;
  $\pi \leftarrow \mathcal{A}^W(\mathrm{crs})$;
    ▷ $\Pr[\pi \cap W \setminus S_f \geq 1] \geq \varepsilon$
  $S_\mathrm{f} \coloneqq S_\mathrm{f} \cup (\pi \cap W)$;
return S_f;

Knowledge Extraction in CRS model

- Knowledge extraction via rewinding:
  - suppose $\mathcal{A}^W(\mathrm{crs})$ succeeds with probability $2^{-\lambda} + \varepsilon \Longrightarrow$ extract with probability ε
  - while $|S_\mathrm{f}| \leq n_\mathrm{f}$, $\mathcal{A}^W(\mathrm{crs})$ outputs weight-1 elements outside of S_f with probability ε
- We achieve:
  - security when weight function is fixed in advance
  - can get security when weight function is not fixed in advance if PRF key is chosen by RO

Algorithm $\mathsf{Extract}^W(\mathcal{A})$

$S_\mathrm{f} := \emptyset$;
while $|S_\mathrm{f}| \leq n_\mathrm{f}$ **do**
  $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$;
  $\pi \leftarrow \mathcal{A}^W(\mathrm{crs})$;
    ▷ $\Pr[\pi \cap W \setminus S_f \geq 1] \geq \varepsilon$
  $S_\mathrm{f} := S_\mathrm{f} \cup (\pi \cap W)$;
return S_f;

Knowledge Extraction in CRS model

- Knowledge extraction via rewinding:
  - suppose $\mathcal{A}^W(\mathrm{crs})$ succeeds with probability $2^{-\lambda} + \varepsilon \Longrightarrow$ extract with probability ε
  - while $|S_f| \leq n_f$, $\mathcal{A}^W(\mathrm{crs})$ outputs weight-1 elements outside of S_f with probability ε
- We achieve:
  - security when weight function is fixed in advance
  - can get security when weight function is not fixed in advance if PRF key is chosen by RO

Algorithm $\mathsf{Extract}^W(\mathcal{A})$

$S_f := \emptyset$;
while $|S_f| \leq n_f$ **do**
  $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$;
  $\pi \leftarrow \mathcal{A}^W(\mathrm{crs})$;
  ▷ $\Pr[\pi \cap W \setminus S_f \geq 1] \geq \varepsilon$
  $S_f := S_f \cup (\pi \cap W)$;
return S_f;

Knowledge Extraction in CRS model

- Knowledge extraction via rewinding:
  - suppose $\mathcal{A}^W(\mathrm{crs})$ succeeds with probability $2^{-\lambda} + \varepsilon \Longrightarrow$ extract with probability ε
  - while $|S_\mathrm{f}| \leq n_\mathrm{f}$, $\mathcal{A}^W(\mathrm{crs})$ outputs weight-1 elements outside of S_f with probability ε
- We achieve:
  - security when weight function is fixed in advance
  - can get security when weight function is not fixed in advance if PRF key is chosen by RO

Algorithm $\mathsf{Extract}^W(\mathcal{A})$

$S_\mathrm{f} := \emptyset$;
while $|S_\mathrm{f}| \leq n_\mathrm{f}$ **do**
  $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$;
  $\pi \leftarrow \mathcal{A}^W(\mathrm{crs})$;
    ▷ $\Pr[\pi \cap W \setminus S_f \geq 1] \geq \varepsilon$
  $S_\mathrm{f} := S_\mathrm{f} \cup (\pi \cap W)$;
return S_f;

Knowledge Extraction in CRS model

- Knowledge extraction via rewinding:
  - suppose $\mathcal{A}^W(\mathrm{crs})$ succeeds with probability $2^{-\lambda} + \varepsilon \implies$ extract with probability ε
  - while $|S_\mathrm{f}| \leq n_\mathrm{f}$, $\mathcal{A}^W(\mathrm{crs})$ outputs weight-1 elements outside of S_f with probability ε
- We achieve:
  - security when weight function is fixed in advance
  - can get security when weight function is not fixed in advance if PRF key is chosen by RO

Algorithm $\mathrm{Extract}^W(\mathcal{A})$

$S_\mathrm{f} \coloneqq \emptyset$;
while $|S_\mathrm{f}| \leq n_\mathrm{f}$ **do**
  $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$;
  $\pi \leftarrow \mathcal{A}^W(\mathrm{crs})$;
    ▷ $\Pr[\pi \cap W \setminus S_f \geq 1] \geq \varepsilon$
  $S_\mathrm{f} \coloneqq S_\mathrm{f} \cup (\pi \cap W)$;
return S_f;

Knowledge Extraction in CRS model

- Knowledge extraction via rewinding:
  - suppose $\mathcal{A}^W(\mathrm{crs})$ succeeds with probability $2^{-\lambda} + \varepsilon \Longrightarrow$ extract with probability ε
  - while $|S_\mathrm{f}| \leq n_\mathrm{f}$, $\mathcal{A}^W(\mathrm{crs})$ outputs weight-1 elements outside of S_f with probability ε
- We achieve:
  - security when weight function is fixed in advance
  - can get security when weight function is not fixed in advance if PRF key is chosen by RO

Algorithm $\mathsf{Extract}^W(\mathcal{A})$

$S_\mathrm{f} \coloneqq \emptyset$;
while $|S_\mathrm{f}| \leq n_\mathrm{f}$ **do**
  $\mathrm{crs} \leftarrow \mathsf{GenCRS}()$;
  $\pi \leftarrow \mathcal{A}^W(\mathrm{crs})$;
    ▷ $\Pr[\pi \cap W \setminus S_f \geq 1] \geq \varepsilon$
  $S_\mathrm{f} \coloneqq S_\mathrm{f} \cup (\pi \cap W)$;
return S_f;