Preserving Randomness for Adaptive Algorithms

William M. Hoza Adam R. Klivans

The University of Texas at Austin

May 25, 2017

Caltech Theory of Computing Seminar

Randomized estimation algorithms

- Algorithm Est(C) estimates some value $\mu(C) \in \mathbb{R}^d$

Randomized estimation algorithms

- Algorithm Est(C) estimates some value $\mu(C) \in \mathbb{R}^d$

$$\Pr[\|\text{Est}(C) - \mu(C)\|_\infty > \varepsilon] \leq \delta$$

Randomized estimation algorithms

- Algorithm $\mathsf{Est}(C)$ estimates some value $\mu(C) \in \mathbb{R}^d$

$$\Pr[\|\mathsf{Est}(C) - \mu(C)\|_\infty > \varepsilon] \leq \delta$$

- Canonical example:

Randomized estimation algorithms

- Algorithm Est(C) estimates some value $\mu(C) \in \mathbb{R}^d$

$$\Pr[\|\mathrm{Est}(C) - \mu(C)\|_\infty > \varepsilon] \leq \delta$$

- Canonical example:
  - C is a Boolean circuit

Randomized estimation algorithms

- Algorithm $\mathrm{Est}(C)$ estimates some value $\mu(C) \in \mathbb{R}^d$

$$\Pr[\|\mathrm{Est}(C) - \mu(C)\|_\infty > \varepsilon] \leq \delta$$

- Canonical example:
  - C is a Boolean circuit
  - $\mu(C) \stackrel{\text{def}}{=} \Pr_x[C(x) = 1] \quad (d = 1)$

Randomized estimation algorithms

- Algorithm Est(C) estimates some value $\mu(C) \in \mathbb{R}^d$

$$\Pr[\|\mathrm{Est}(C) - \mu(C)\|_\infty > \varepsilon] \leq \delta$$

- Canonical example:
  - C is a Boolean circuit
  - $\mu(C) \stackrel{\text{def}}{=} \Pr_x[C(x) = 1] \quad (d = 1)$
  - Est(C) evaluates C at several randomly chosen points

Executing Est many times

Executing Est many times

- Goal: Execute $\text{Est}(C_1), \text{Est}(C_2), \ldots, \text{Est}(C_k)$

Executing Est many times

- Goal: Execute $\text{Est}(C_1), \text{Est}(C_2), \ldots, \text{Est}(C_k)$
- Say Est uses n random bits

Executing Est many times

- Goal: Execute $\text{Est}(C_1), \text{Est}(C_2), \ldots, \text{Est}(C_k)$
- Say Est uses n random bits
- Naïve implementation: nk random bits

Executing Est many times

- Goal: Execute $\mathrm{Est}(C_1), \mathrm{Est}(C_2), \ldots, \mathrm{Est}(C_k)$
- Say Est uses n random bits
- Naïve implementation: nk random bits
- **Can we do better?**

Executing Est many times

- Goal: Execute $\text{Est}(C_1), \text{Est}(C_2), \ldots, \text{Est}(C_k)$
- Say Est uses n random bits
- Naïve implementation: nk random bits
- **Can we do better?**

- **Theorem**: Can use just $n + O(k \log(d+1))$ random bits!

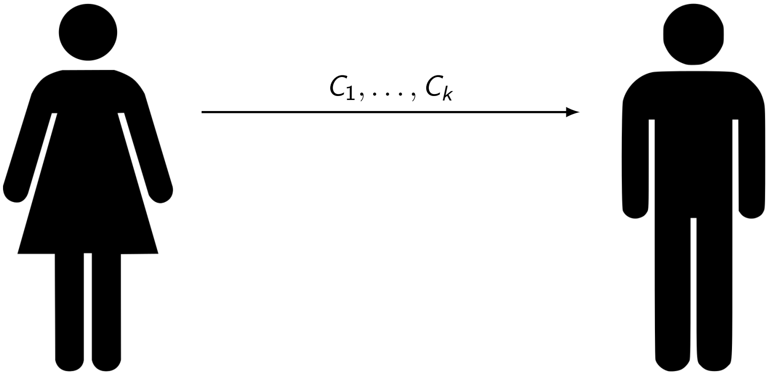
Executing Est many times

- Goal: Execute $\mathrm{Est}(C_1), \mathrm{Est}(C_2), \ldots, \mathrm{Est}(C_k)$
- Say Est uses n random bits
- Naïve implementation: nk random bits
- **Can we do better?**

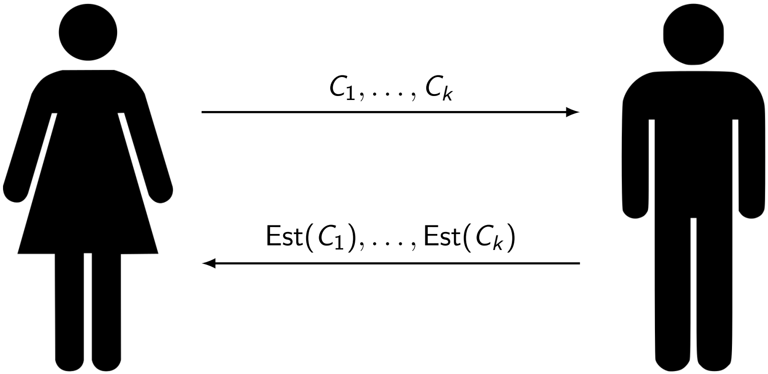
- **Theorem**: Can use just $n + O(k \log(d+1))$ random bits!
  - Slight increases in error, failure probability

Nonadaptive setting

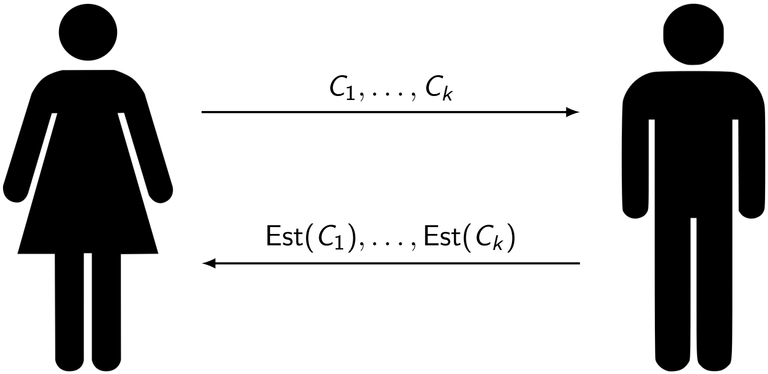
Nonadaptive setting



Nonadaptive setting

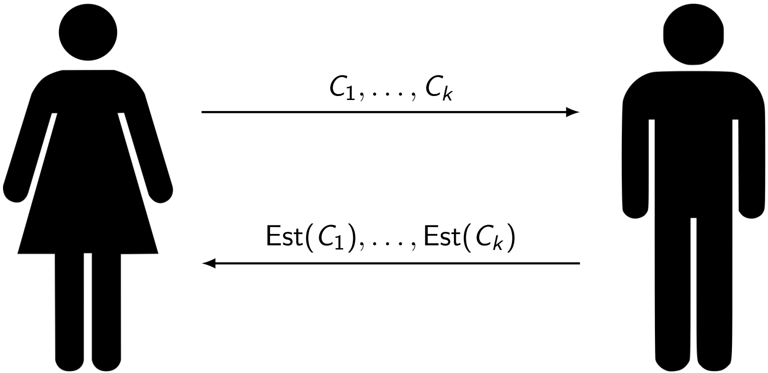


Nonadaptive setting



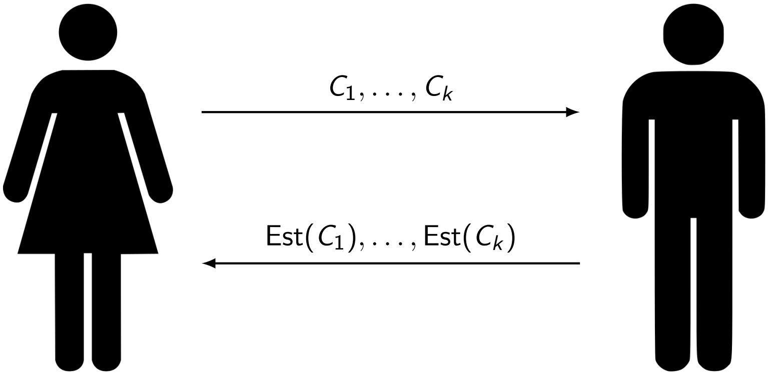
- Algorithm that uses just n random bits:

Nonadaptive setting



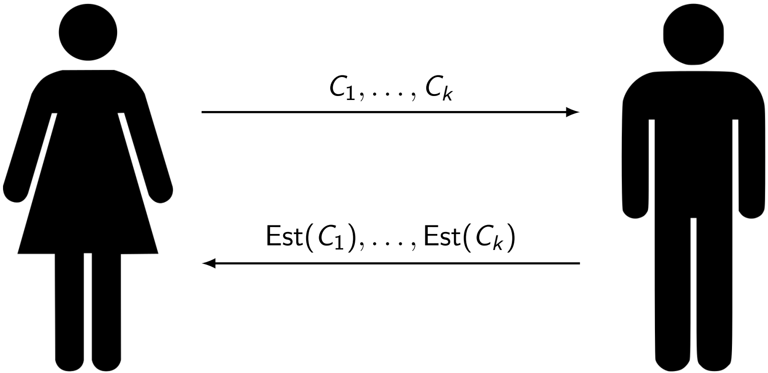
- Algorithm that uses just n random bits:
  1. Pick $X \in \{0,1\}^n$ uniformly at random once

Nonadaptive setting



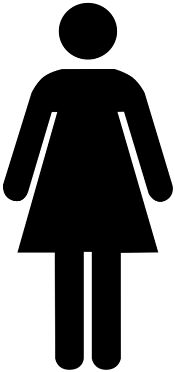
- Algorithm that uses just n random bits:
  1. Pick $X \in \{0,1\}^n$ uniformly at random once
  2. Execute $\mathrm{Est}(C_1, X), \mathrm{Est}(C_2, X), \ldots, \mathrm{Est}(C_k, X)$

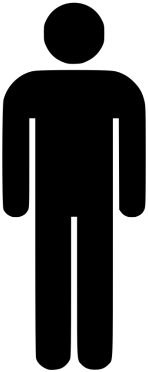
Nonadaptive setting



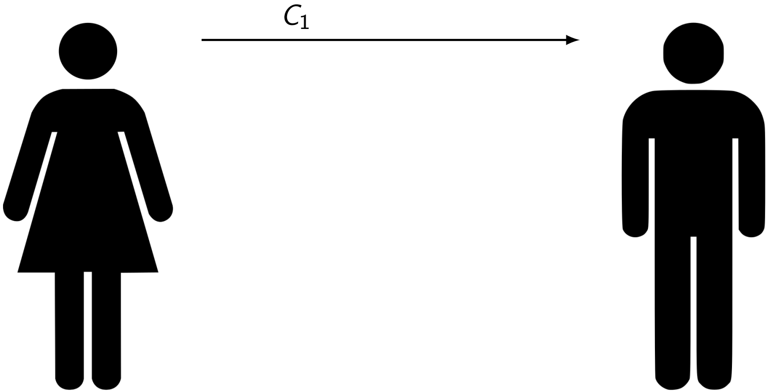
- Algorithm that uses just n random bits:
  1. Pick $X \in \{0,1\}^n$ uniformly at random once
  2. Execute $\text{Est}(C_1, X), \text{Est}(C_2, X), \ldots, \text{Est}(C_k, X)$
- Overall failure probability is still $k\delta$ (union bound)

Adaptive setting

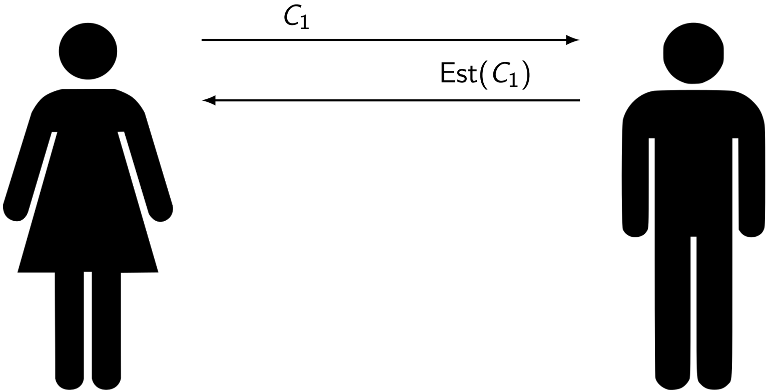




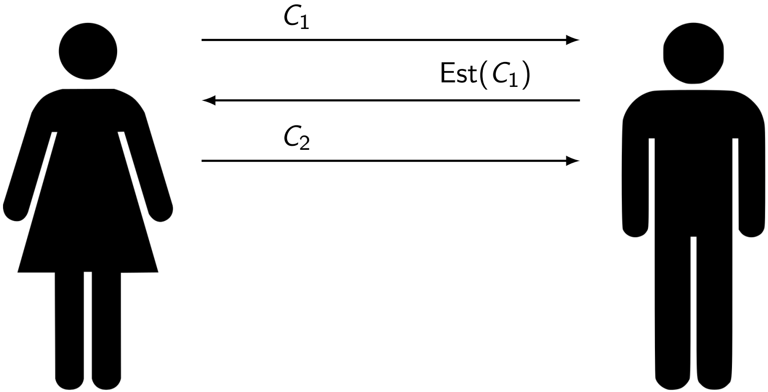
Adaptive setting



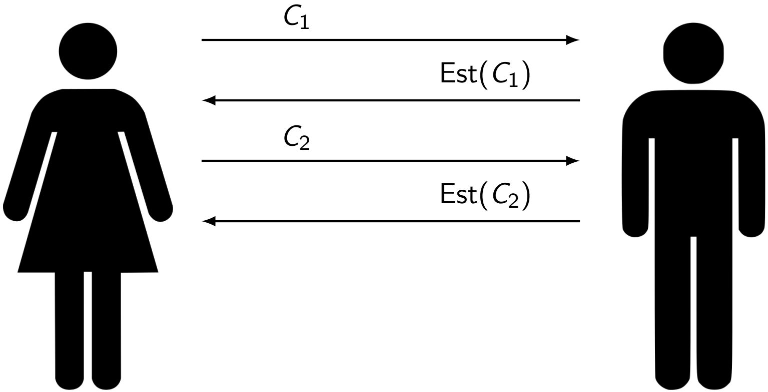
Adaptive setting



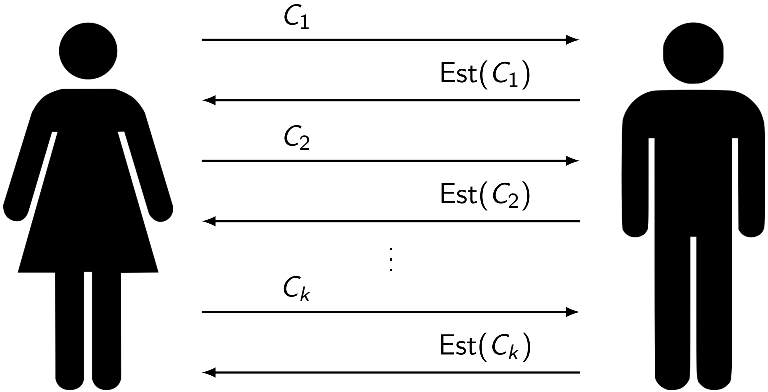
Adaptive setting



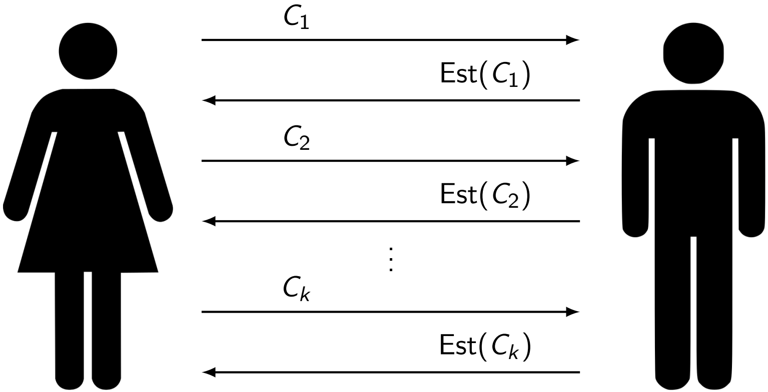
Adaptive setting



Adaptive setting

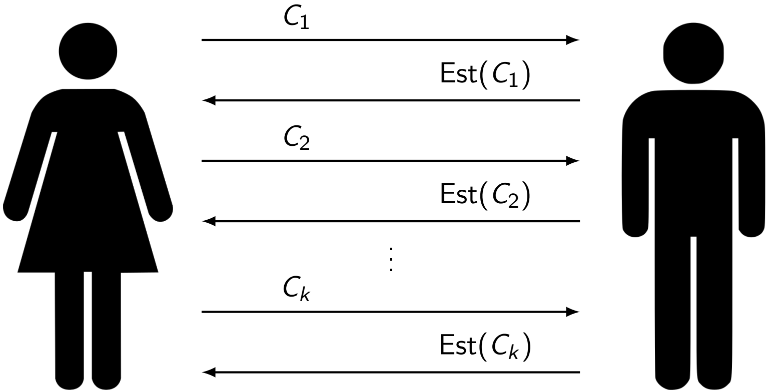


Adaptive setting



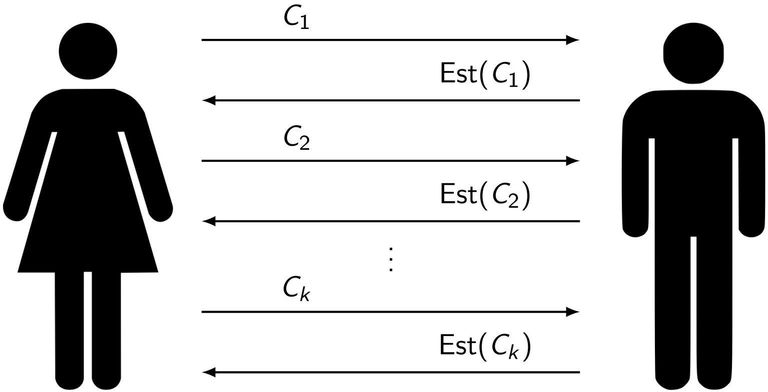
- Let X be the randomness used for $\text{Est}(C_1)$

Adaptive setting



- Let X be the randomness used for $\text{Est}(C_1)$
- C_2 is stochastically dependent on X

Adaptive setting



- Let X be the randomness used for $\mathrm{Est}(C_1)$
- C_2 is stochastically dependent on X
- Failure probability of $\mathrm{Est}(C_2, X)$ is ???

Concentrated functions

- $f : \{0,1\}^n \rightarrow \mathbb{R}^d$

Concentrated functions

- $f : \{0,1\}^n \to \mathbb{R}^d$
- **Definition**: f is (ε, δ)-concentrated at $\mu \in \mathbb{R}^d$ if

$$\Pr_X[\|f(X) - \mu\|_\infty > \varepsilon] \leq \delta.$$

Concentrated functions

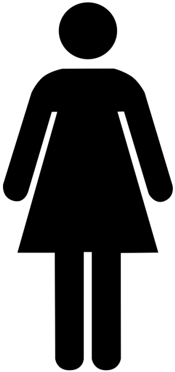
- $f : \{0,1\}^n \to \mathbb{R}^d$
- **Definition**: f is (ε, δ)-concentrated at $\mu \in \mathbb{R}^d$ if

$$\Pr_X[\|f(X) - \mu\|_\infty > \varepsilon] \leq \delta.$$

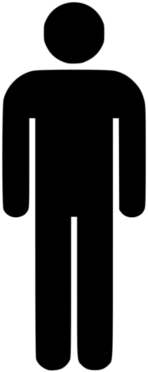
- Example: $f(X) \stackrel{\text{def}}{=} \text{Est}(C, X)$

Randomness steward model

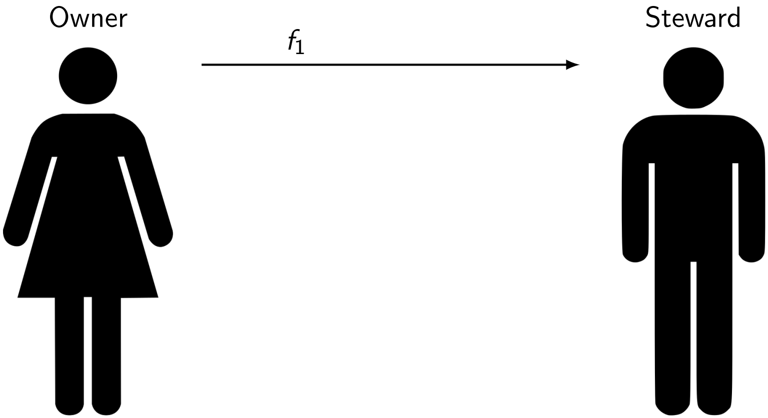
Owner



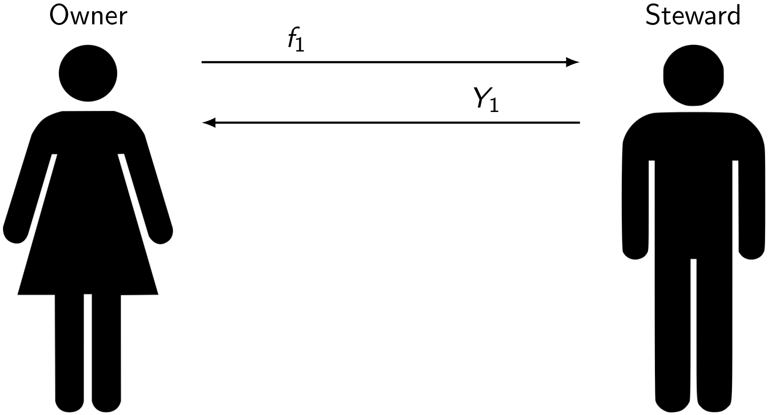
Steward



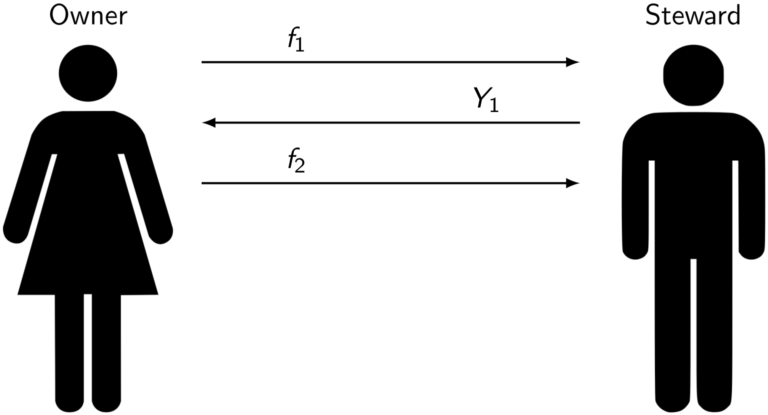
Randomness steward model



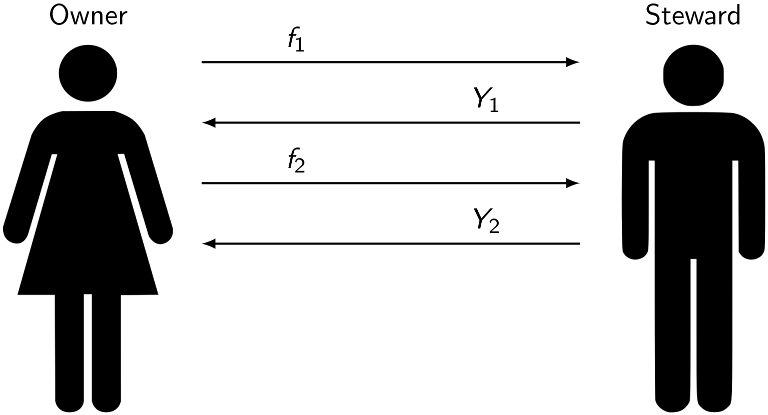
Randomness steward model



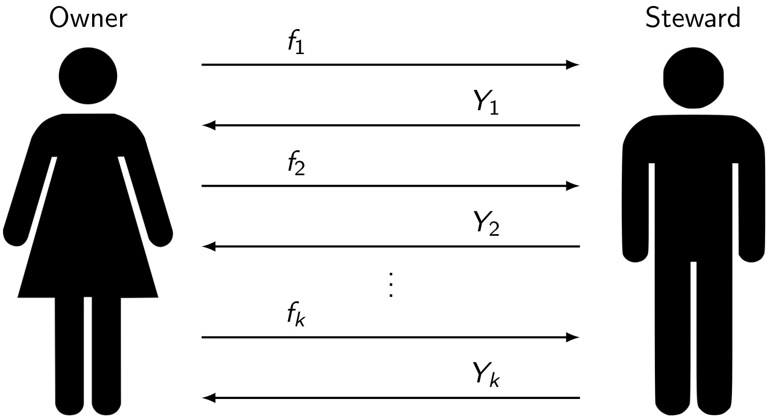
Randomness steward model



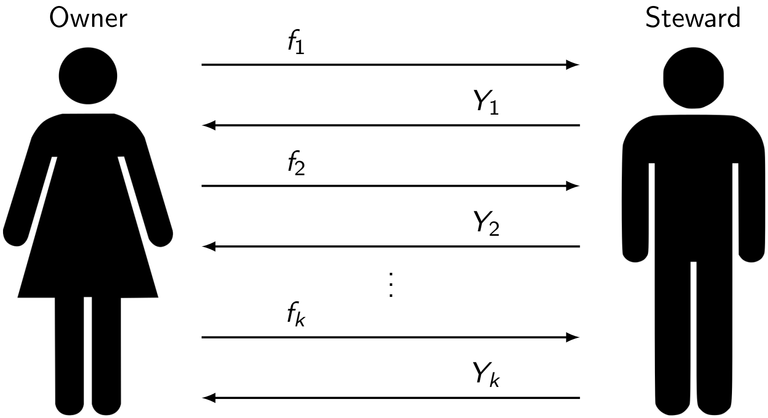
Randomness steward model



Randomness steward model

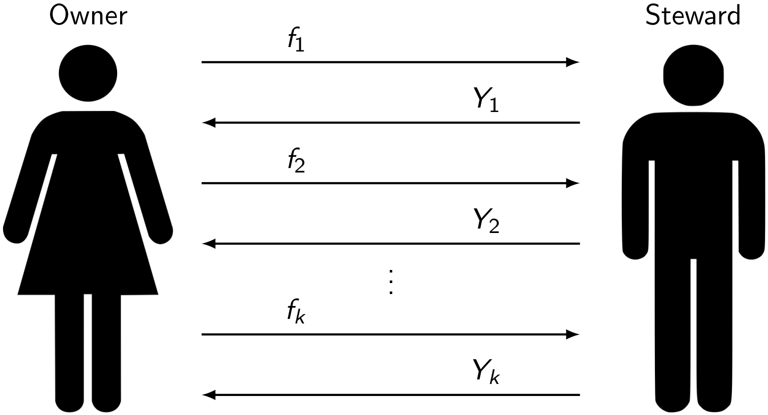


Randomness steward model



- Each f_i is (ε, δ)-concentrated at some μ_i

Randomness steward model



- Each f_i is (ε, δ)-concentrated at some μ_i
- Steward requirement: For any owner,

$$\Pr\left[\max_i \| Y_i - \mu_i \|_\infty > \varepsilon'\right] \leq \delta'$$

One-query stewards

- **Definition**: One-query steward: Only accesses each f_i by querying a single point $f_i(X_i)$

One-query stewards

- **Definition**: One-query steward: Only accesses each f_i by querying a single point $f_i(X_i)$
  - Querying f_i corresponds to executing Est

One-query stewards

- **Definition**: One-query steward: Only accesses each f_i by querying a single point $f_i(X_i)$
  - Querying f_i corresponds to executing Est
  - The owner does not see X_i

Warm-up steward

- **Theorem**: For any $n, k, \varepsilon, \delta$, there exists a one-query steward for $d = 1$ with

Warm-up steward

- **Theorem**: For any $n, k, \varepsilon, \delta$, there exists a one-query steward for $d = 1$ with
  - Error $\varepsilon' \leq O(\varepsilon)$ (vs. naïve ε)

Warm-up steward

- **Theorem**: For any $n, k, \varepsilon, \delta$, there exists a one-query steward for $d = 1$ with
  - Error $\varepsilon' \leq O(\varepsilon)$ (vs. naïve ε)
  - Failure probability $\delta' \leq 2^k \cdot \delta$ (vs. naïve $k\delta$)

Warm-up steward

- **Theorem**: For any $n, k, \varepsilon, \delta$, there exists a one-query steward for $d = 1$ with
  - Error $\varepsilon' \leq O(\varepsilon)$ (vs. naïve ε)
  - Failure probability $\delta' \leq 2^k \cdot \delta$ (vs. naïve $k\delta$)
  - Randomness n (vs. naïve nk)

Warm-up steward

- **Theorem**: For any $n, k, \varepsilon, \delta$, there exists a one-query steward for $d = 1$ with
  - Error $\varepsilon' \leq O(\varepsilon)$ (vs. naïve ε)
  - Failure probability $\delta' \leq 2^k \cdot \delta$ (vs. naïve $k\delta$)
  - Randomness n (vs. naïve nk)
- The steward: "Reuse randomness and round"

Warm-up steward

- **Theorem**: For any $n, k, \varepsilon, \delta$, there exists a one-query steward for $d = 1$ with
  - Error $\varepsilon' \leq O(\varepsilon)$ (vs. naïve ε)
  - Failure probability $\delta' \leq 2^k \cdot \delta$ (vs. naïve $k\delta$)
  - Randomness n (vs. naïve nk)
- The steward: "Reuse randomness and round"
  - Pick $X \in \{0, 1\}^n$ uniformly at random once

Warm-up steward

- **Theorem**: For any $n, k, \varepsilon, \delta$, there exists a one-query steward for $d = 1$ with
  - Error $\varepsilon' \leq O(\varepsilon)$ (vs. naïve ε)
  - Failure probability $\delta' \leq 2^k \cdot \delta$ (vs. naïve $k\delta$)
  - Randomness n (vs. naïve nk)
- The steward: "Reuse randomness and round"
  - Pick $X \in \{0, 1\}^n$ uniformly at random once
  - For $i = 1$ to k: Return $f_i(X)$, rounded to multiple of 2ε

Analysis of warm-up steward

Analysis of warm-up steward

Analysis of warm-up steward

Analysis of warm-up steward

Analysis of warm-up steward

- Imagine if the steward always returns $A(\mu)$ or $B(\mu)$...

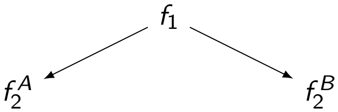
Analysis of warm-up steward

- Imagine if the steward always returns $A(\mu)$ or $B(\mu)$...

$$f_1$$

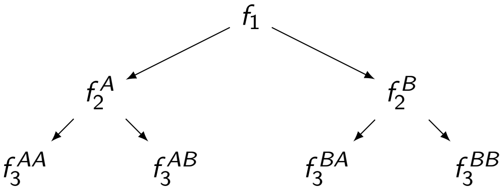
Analysis of warm-up steward

- Imagine if the steward always returns $A(\mu)$ or $B(\mu)$...



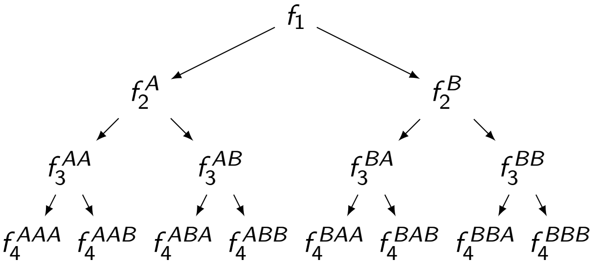
Analysis of warm-up steward

- Imagine if the steward always returns $A(\mu)$ or $B(\mu)$...



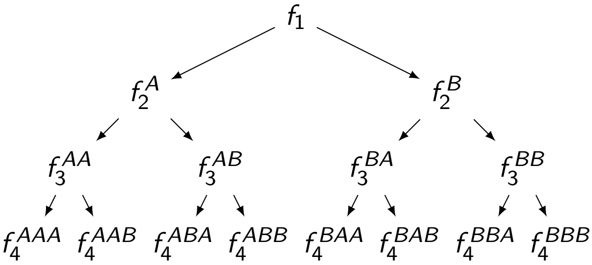
Analysis of warm-up steward

- Imagine if the steward always returns $A(\mu)$ or $B(\mu)$...



Analysis of warm-up steward

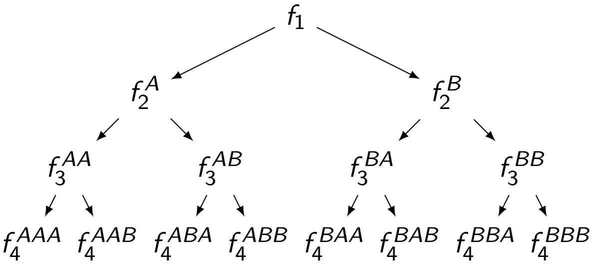
- Imagine if the steward always returns $A(\mu)$ or $B(\mu)$...



- Union bound: $\Pr[X$ good for every function in tree$] \geq 1 - 2^k\delta$

Analysis of warm-up steward

- Imagine if the steward always returns $A(\mu)$ or $B(\mu)$...



- Union bound: $\Pr[X$ good for every function in tree$] \geq 1 - 2^k\delta$
- If so, inductively, every f_i is in the tree!

Main result

- **Theorem**: For all $n, k, d, \varepsilon, \delta, \gamma$, there is an efficient one-query steward with

Main result

- **Theorem**: For all $n, k, d, \varepsilon, \delta, \gamma$, there is an efficient one-query steward with
  - Error $\varepsilon' \leq O(\varepsilon d)$

Main result

- **Theorem**: For all $n, k, d, \varepsilon, \delta, \gamma$, there is an efficient one-query steward with
  - Error $\varepsilon' \leq O(\varepsilon d)$
  - Failure probability $\delta' \leq k\delta + \gamma$

Main result

- **Theorem**: For all $n, k, d, \varepsilon, \delta, \gamma$, there is an efficient one-query steward with
  - Error $\varepsilon' \leq O(\varepsilon d)$
  - Failure probability $\delta' \leq k\delta + \gamma$
  - # random bits $n + O(k \log(d+1) + \log k \log(1/\gamma))$

Main steward

- Pick random seed X, compute $(X_1, \ldots, X_k) = \mathsf{Gen}(X)$

Main steward

- Pick random seed X, compute $(X_1, \ldots, X_k) = \text{Gen}(X)$
- For $i = 1$ to k:

Main steward

- Pick random seed X, compute $(X_1, \ldots, X_k) = \text{Gen}(X)$
- For $i = 1$ to k:
  - Obtain $W_i = f_i(X_i)$

Main steward

- Pick random seed X, compute $(X_1, \ldots, X_k) = \mathsf{Gen}(X)$
- For $i = 1$ to k:
  - Obtain $W_i = f_i(X_i)$
  - Shift and round W_i to determine output Y_i

Main steward

- Pick random seed X, compute $(X_1, \ldots, X_k) = \text{Gen}(X)$
- For $i = 1$ to k:
  - Obtain $W_i = f_i(X_i)$
  - Shift and round W_i to determine output Y_i

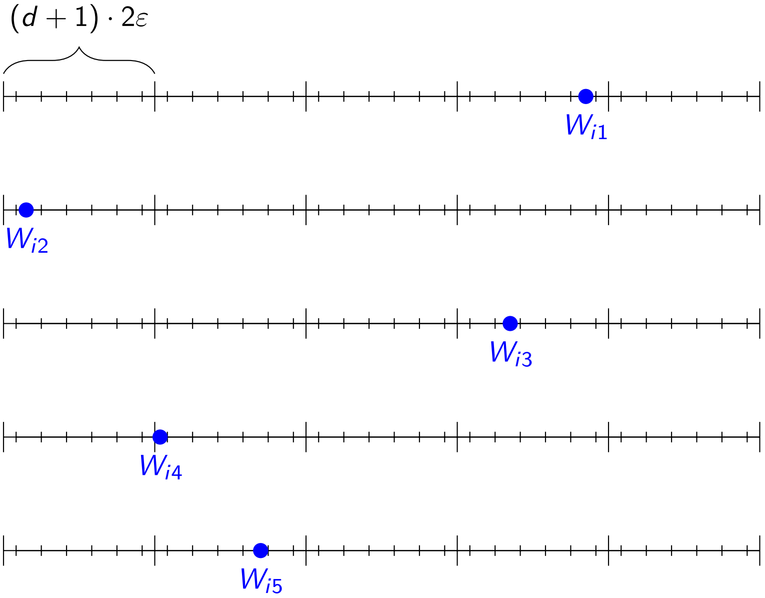
- Ingredient 1: Gen: PRG for block decision trees

Main steward

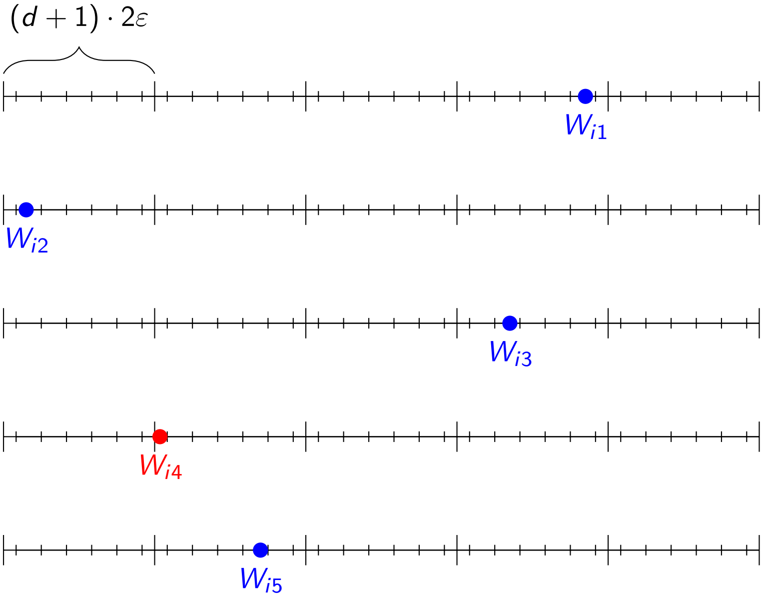
- Pick random seed X, compute $(X_1, \ldots, X_k) = \text{Gen}(X)$
- For $i = 1$ to k:
  - Obtain $W_i = f_i(X_i)$
  - Shift and round W_i to determine output Y_i

- Ingredient 1: Gen: PRG for block decision trees
- Ingredient 2: Deterministic shifting and rounding algorithm

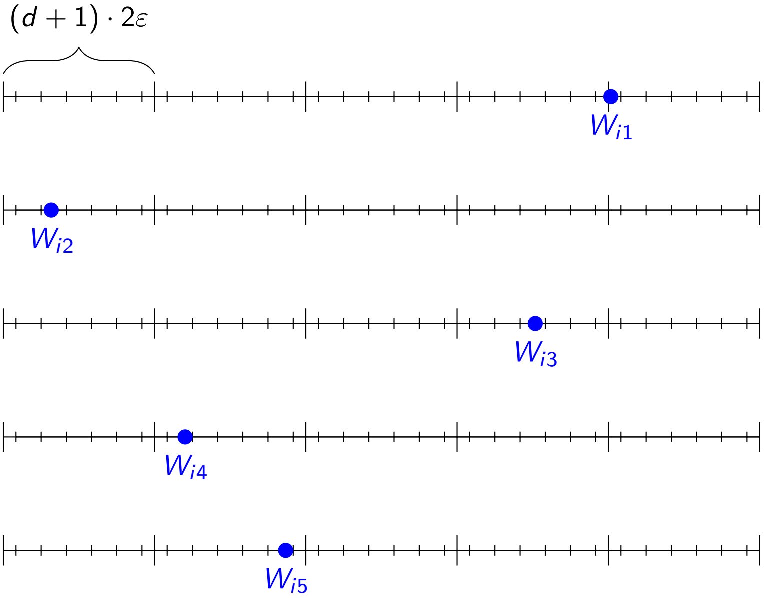
Shifting and rounding algorithm



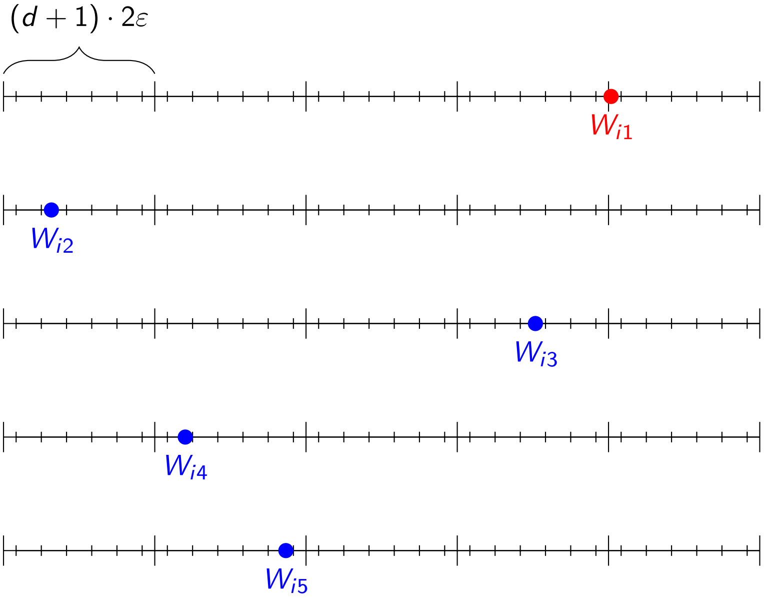
Shifting and rounding algorithm



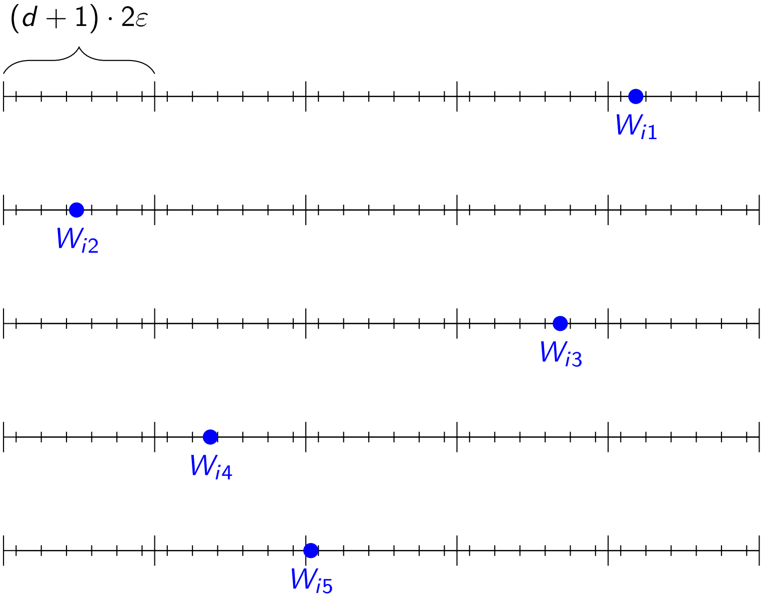
Shifting and rounding algorithm



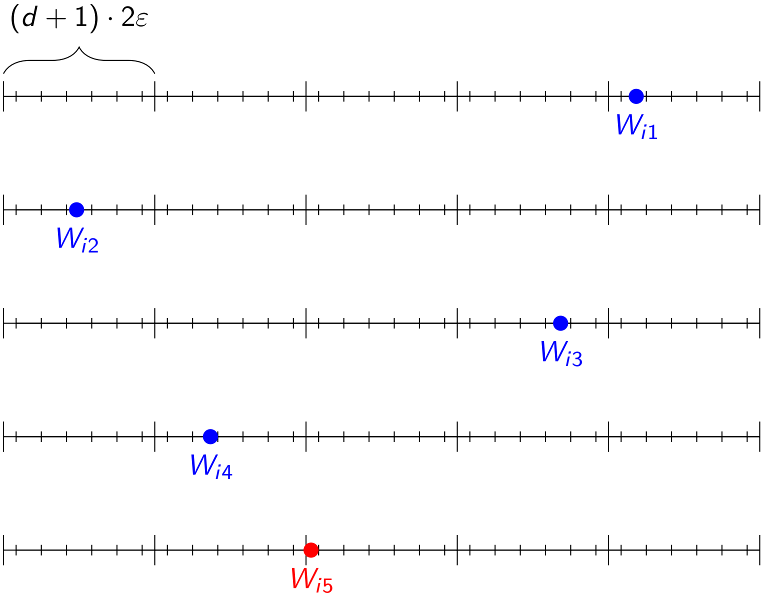
Shifting and rounding algorithm



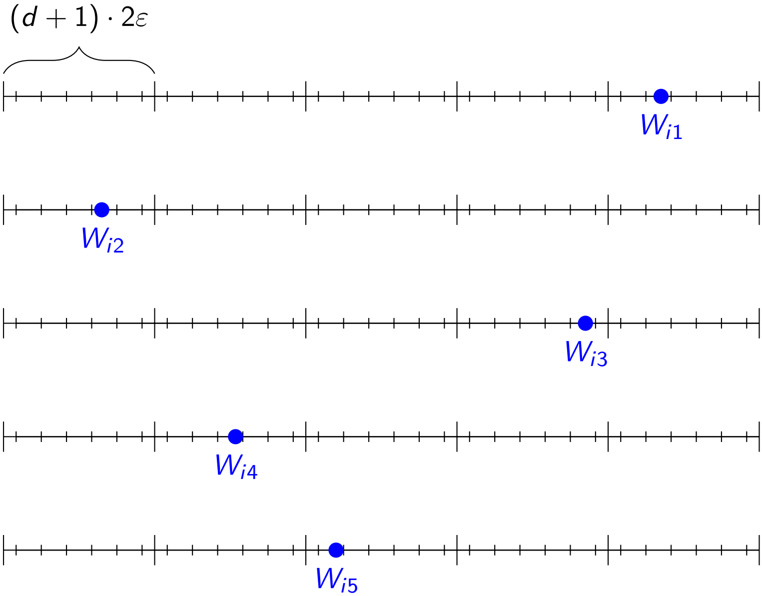
Shifting and rounding algorithm



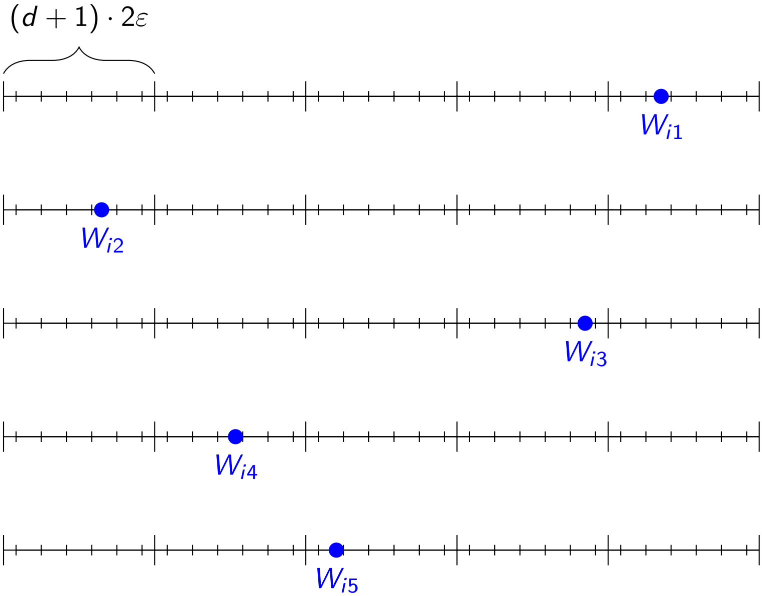
Shifting and rounding algorithm



Shifting and rounding algorithm



Shifting and rounding algorithm



Analysis of shifting and rounding algorithm

- For $W \in \mathbb{R}^d$ and $\Delta \in [d+1]$, define $R_\Delta(W) \in \mathbb{R}^d$ by shifting W according to Δ, then rounding

Analysis of shifting and rounding algorithm

- For $W \in \mathbb{R}^d$ and $\Delta \in [d+1]$, define $R_\Delta(W) \in \mathbb{R}^d$ by shifting W according to Δ, then rounding
- By construction, $Y_i = R_\Delta(W_i)$ for some Δ

Analysis of shifting and rounding algorithm

- For $W \in \mathbb{R}^d$ and $\Delta \in [d+1]$, define $R_\Delta(W) \in \mathbb{R}^d$ by shifting W according to Δ, then rounding
- By construction, $Y_i = R_\Delta(W_i)$ for some Δ
- Imagine if $Y_i = R_\Delta(\mu_i)$ for some Δ...

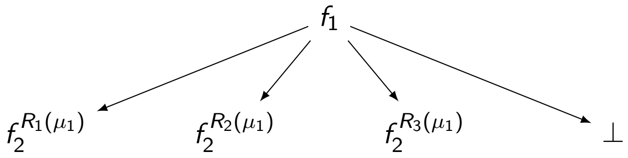
Analysis of shifting and rounding algorithm

- For $W \in \mathbb{R}^d$ and $\Delta \in [d+1]$, define $R_\Delta(W) \in \mathbb{R}^d$ by shifting W according to Δ, then rounding
- By construction, $Y_i = R_\Delta(W_i)$ for some Δ
- Imagine if $Y_i = R_\Delta(\mu_i)$ for some Δ...

f_1

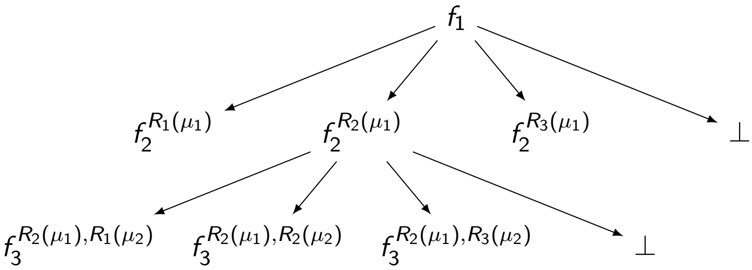
Analysis of shifting and rounding algorithm

- For $W \in \mathbb{R}^d$ and $\Delta \in [d+1]$, define $R_\Delta(W) \in \mathbb{R}^d$ by shifting W according to Δ, then rounding
- By construction, $Y_i = R_\Delta(W_i)$ for some Δ
- Imagine if $Y_i = R_\Delta(\mu_i)$ for some Δ...

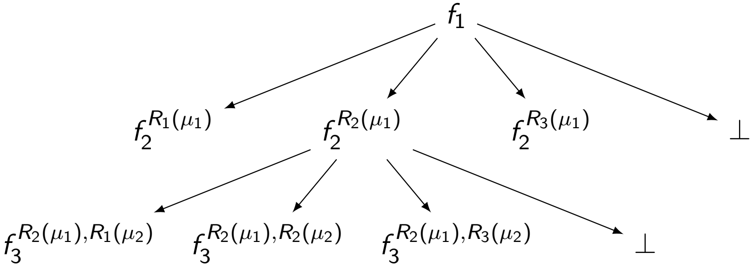


Analysis of shifting and rounding algorithm

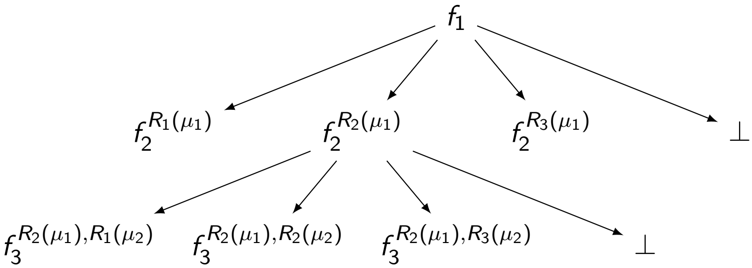
- For $W \in \mathbb{R}^d$ and $\Delta \in [d+1]$, define $R_\Delta(W) \in \mathbb{R}^d$ by shifting W according to Δ, then rounding
- By construction, $Y_i = R_\Delta(W_i)$ for some Δ
- Imagine if $Y_i = R_\Delta(\mu_i)$ for some Δ...



Certification tree

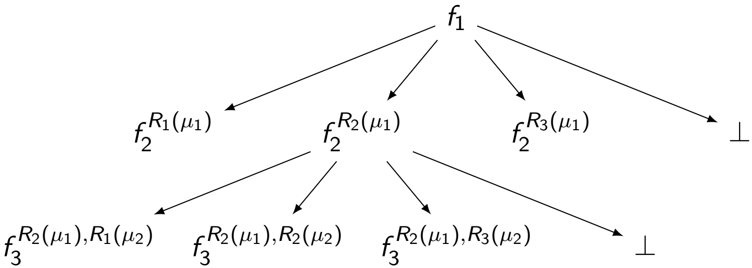


Certification tree



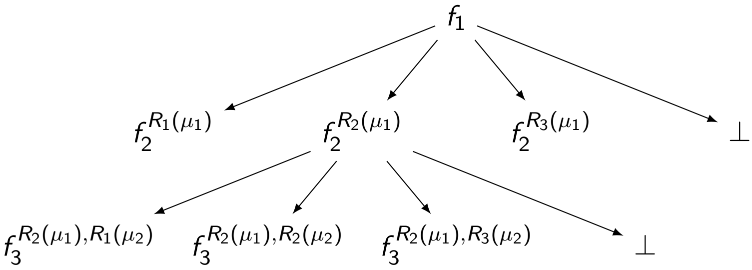
- A sequence $(X_1, \ldots, X_k)$ of query points determines:

Certification tree



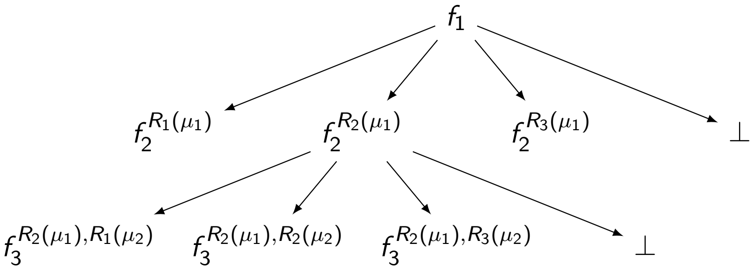
- A sequence $(X_1, \ldots, X_k)$ of query points determines:
  - A transcript $(f_1, Y_1, f_2, Y_2, \ldots, f_k, Y_k)$

Certification tree



- A sequence $(X_1, \ldots, X_k)$ of query points determines:
  - A transcript $(f_1, Y_1, f_2, Y_2, \ldots, f_k, Y_k)$
  - A path P through tree

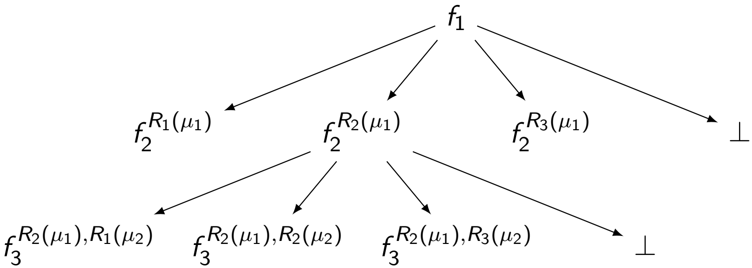
Certification tree



- A sequence $(X_1, \ldots, X_k)$ of query points determines:
  - A transcript $(f_1, Y_1, f_2, Y_2, \ldots, f_k, Y_k)$
  - A path P through tree
- If we pick $X_1, \ldots, X_k$ independently and u.a.r.,

$$\Pr_{(X_1,\ldots,X_k)}[P \text{ has a } \perp \text{ node}] \leq k\delta$$

Certification tree



- A sequence $(X_1, \ldots, X_k)$ of query points determines:
  - A transcript $(f_1, Y_1, f_2, Y_2, \ldots, f_k, Y_k)$
  - A path P through tree
- If we pick $X_1, \ldots, X_k$ independently and u.a.r.,

$$\Pr_{(X_1,\ldots,X_k)}[P \text{ has a } \perp \text{ node}] \leq k\delta$$

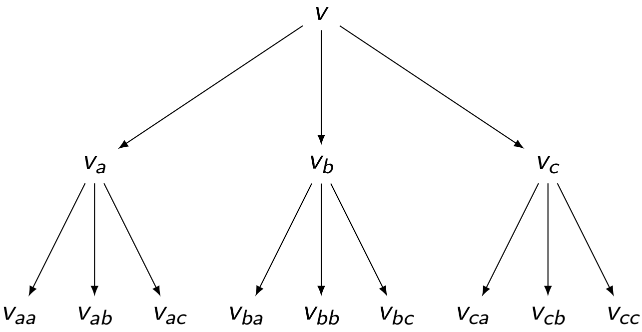
- (Certification) No $\perp$ nodes in $P \implies$ every Y_i has error $O(\varepsilon d)$

Block decision trees

- (k, n, q) block decision tree: Full q-ary tree of height k

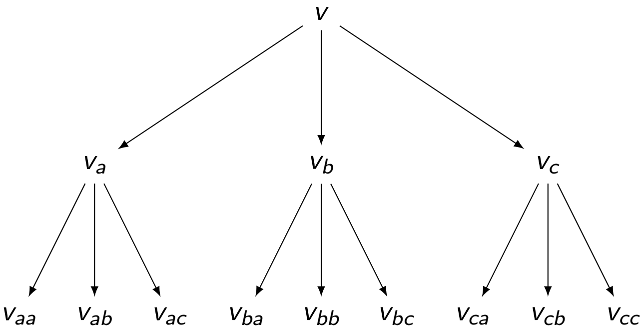
Block decision trees

- (k, n, q) block decision tree: Full q-ary tree of height k



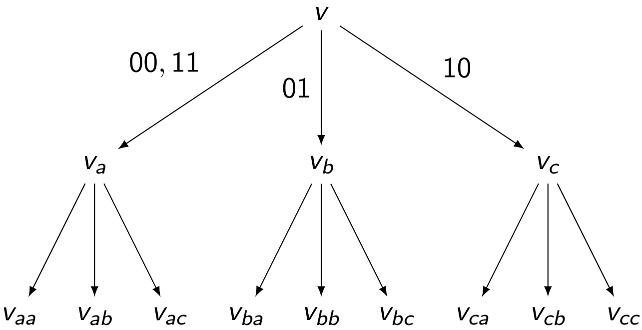
Block decision trees

- (k, n, q) block decision tree: Full q-ary tree of height k
- Each internal node v_s has a function $v_s : \{0,1\}^n \to [q]$



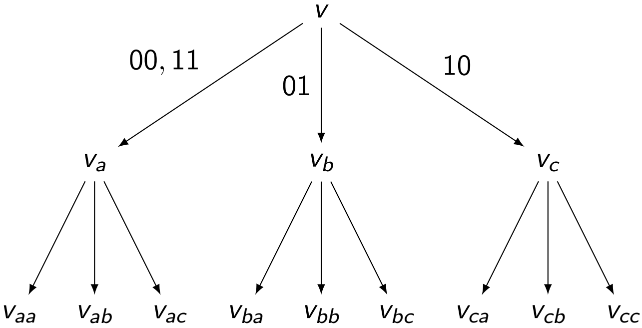
Block decision trees

- (k, n, q) block decision tree: Full q-ary tree of height k
- Each internal node v_s has a function $v_s : \{0,1\}^n \to [q]$



Block decision trees

- (k, n, q) block decision tree: Full q-ary tree of height k
- Each internal node v_s has a function $v_s : \{0,1\}^n \to [q]$
- Tree reads nk bits and outputs a leaf



PRG for block decision trees

- **Theorem**: There is an efficient γ-PRG for block decision trees with seed length

$$n + O(k \log q + \log k \log(1/\gamma))$$

PRG for block decision trees

- **Theorem**: There is an efficient γ-PRG for block decision trees with seed length

$$n + O(k \log q + \log k \log(1/\gamma))$$

- Proof idea: Modify parameters of INW generator

PRG for block decision trees

- **Theorem**: There is an efficient γ-PRG for block decision trees with seed length

$$n + O(k \log q + \log k \log(1/\gamma))$$

- Proof idea: Modify parameters of INW generator
- This generator fools the certification tree

PRG for block decision trees

- **Theorem**: There is an efficient γ-PRG for block decision trees with seed length

$$n + O(k \log q + \log k \log(1/\gamma))$$

- Proof idea: Modify parameters of INW generator
- This generator fools the certification tree
- No need to fool steward/owner protocol! $\square$

Application: Randomness-efficient Goldreich-Levin

Application: Randomness-efficient Goldreich-Levin

- Oracle access to $x \in \{0,1\}^{2^n}$

Application: Randomness-efficient Goldreich-Levin

- Oracle access to $x \in \{0,1\}^{2^n}$
- **Theorem**: Can find all Hadamard codewords that agree with x in $(\frac{1}{2} + \theta)$-fraction of positions

Application: Randomness-efficient Goldreich-Levin

- Oracle access to $x \in \{0,1\}^{2^n}$
- **Theorem**: Can find all Hadamard codewords that agree with x in $(\frac{1}{2} + \theta)$-fraction of positions
  - Runtime $\text{poly}(n, 1/\theta, \log(1/\delta))$ $\quad$ ($\delta =$ failure prob)

Application: Randomness-efficient Goldreich-Levin

- Oracle access to $x \in \{0,1\}^{2^n}$
- **Theorem**: Can find all Hadamard codewords that agree with x in $(\frac{1}{2}+\theta)$-fraction of positions
  - Runtime $\text{poly}(n, 1/\theta, \log(1/\delta))$ ($\delta =$ failure prob)
  - $O(n + \log n \log(1/\delta))$ random bits (independent of θ!)

Application: Randomness-efficient Goldreich-Levin

- Oracle access to $x \in \{0,1\}^{2^n}$
- **Theorem**: Can find all Hadamard codewords that agree with x in $(\frac{1}{2}+\theta)$-fraction of positions
  - Runtime $\mathrm{poly}(n, 1/\theta, \log(1/\delta))$ ($\delta =$ failure prob)
  - $O(n + \log n \log(1/\delta))$ random bits (independent of θ!)
- Previous best: $O(n \log(n/\theta) \log(1/(\delta\theta)))$ random bits (Bshouty et al. '04)

Application: Randomness-efficient Goldreich-Levin

- Oracle access to $x \in \{0,1\}^{2^n}$
- **Theorem**: Can find all Hadamard codewords that agree with x in $(\frac{1}{2} + \theta)$-fraction of positions
  - Runtime $\mathrm{poly}(n, 1/\theta, \log(1/\delta))$ ($\delta =$ failure prob)
  - $O(n + \log n \log(1/\delta))$ random bits (independent of θ!)
- Previous best: $O(n \log(n/\theta) \log(1/(\delta\theta)))$ random bits (Bshouty et al. '04)
- Proof ingredients:

Application: Randomness-efficient Goldreich-Levin

- Oracle access to $x \in \{0,1\}^{2^n}$
- **Theorem**: Can find all Hadamard codewords that agree with x in $(\frac{1}{2} + \theta)$-fraction of positions
  - Runtime $\text{poly}(n, 1/\theta, \log(1/\delta))$ $(\delta = \text{failure prob})$
  - $O(n + \log n \log(1/\delta))$ random bits (independent of θ!)
- Previous best: $O(n \log(n/\theta) \log(1/(\delta\theta)))$ random bits (Bshouty et al. '04)
- Proof ingredients:
  - Standard Goldreich-Levin algorithm

Application: Randomness-efficient Goldreich-Levin

- Oracle access to $x \in \{0,1\}^{2^n}$
- **Theorem**: Can find all Hadamard codewords that agree with x in $(\frac{1}{2} + \theta)$-fraction of positions
  - Runtime $\text{poly}(n, 1/\theta, \log(1/\delta))$ $(\delta = \text{failure prob})$
  - $O(n + \log n \log(1/\delta))$ random bits (independent of θ!)
- Previous best: $O(n \log(n/\theta) \log(1/(\delta\theta)))$ random bits (Bshouty et al. '04)
- Proof ingredients:
  - Standard Goldreich-Levin algorithm
  - Our steward with $d = \text{poly}(1/\theta)$

Application: Randomness-efficient Goldreich-Levin

- Oracle access to $x \in \{0,1\}^{2^n}$
- **Theorem**: Can find all Hadamard codewords that agree with x in $(\frac{1}{2} + \theta)$-fraction of positions
  - Runtime $\text{poly}(n, 1/\theta, \log(1/\delta))$ $(\delta = \text{failure prob})$
  - $O(n + \log n \log(1/\delta))$ random bits (independent of θ!)
- Previous best: $O(n \log(n/\theta) \log(1/(\delta\theta)))$ random bits (Bshouty et al. '04)
- Proof ingredients:
  - Standard Goldreich-Levin algorithm
  - Our steward with $d = \text{poly}(1/\theta)$
  - Goldreich-Wigderson sampler

Landscape of stewards

ε'	δ'	Randomness complexity	Reference

Landscape of stewards

ε'	δ'	Randomness complexity	Reference
ε	$k\delta$	nk	Naïve

Landscape of stewards

ε'	δ'	Randomness complexity	Reference
ε	$k\delta$	nk	Naïve
$O(\varepsilon)$	$2^k\delta$	n (works for $d = 1$ only)	This work

Landscape of stewards

ε'	δ'	Randomness complexity	Reference
ε	$k\delta$	nk	Naïve
$O(\varepsilon)$	$2^k\delta$	n (works for $d = 1$ only)	This work
$O(\varepsilon d)$	$k\delta + \gamma$	$n + O(k \log(d+1) + \log k \log(1/\gamma))$	This work

Landscape of stewards

- Steward model captures derandomization constructions in literature

ε'	δ'	Randomness complexity	Reference
ε	$k\delta$	nk	Naïve
$O(\varepsilon)$	$2^k\delta$	n (works for $d = 1$ only)	This work
$O(\varepsilon d)$	$k\delta + \gamma$	$n + O(k \log(d+1) + \log k \log(1/\gamma))$	This work

Landscape of stewards

- Steward model captures derandomization constructions in literature

ε'	δ'	Randomness complexity	Reference
ε	$k\delta$	nk	Naïve
$O(\varepsilon)$	$2^k\delta$	n (works for $d = 1$ only)	This work
$O(\varepsilon d)$	$k\delta + \gamma$	$n + O(k \log(d+1) + \log k \log(1/\gamma))$	This work
$O(\varepsilon kd/\gamma)$	$k\delta + \gamma$	$n + O(k \log k + k \log d + k \log(1/\gamma))$	$\approx$ SZ '99

Landscape of stewards

- Steward model captures derandomization constructions in literature

ε'	δ'	Randomness complexity	Reference
ε	$k\delta$	nk	Naïve
$O(\varepsilon)$	$2^k\delta$	n (works for $d = 1$ only)	This work
$O(\varepsilon d)$	$k\delta + \gamma$	$n + O(k\log(d+1) + \log k \log(1/\gamma))$	This work
$O(\varepsilon kd/\gamma)$	$k\delta + \gamma$	$n + O(k\log k + k\log d + k\log(1/\gamma))$	≈ SZ '99
$O(\varepsilon)$	$k\delta + k/2^{n^{\Omega(1)}}$	$O(n^6 + kd)$	≈ IZ '89

Landscape of stewards

- Steward model captures derandomization constructions in literature

ε'	δ'	Randomness complexity	Reference
ε	$k\delta$	nk	Naïve
$O(\varepsilon)$	$2^k\delta$	n (works for $d = 1$ only)	This work
$O(\varepsilon d)$	$k\delta + \gamma$	$n + O(k \log(d+1) + \log k \log(1/\gamma))$	This work
$O(\varepsilon kd/\gamma)$	$k\delta + \gamma$	$n + O(k \log k + k \log d + k \log(1/\gamma))$	$\approx$ SZ '99
$O(\varepsilon)$	$k\delta + k/2^{n^{\Omega(1)}}$	$O(n^6 + kd)$	$\approx$ IZ '89
$O(\varepsilon)$	$k\delta + \gamma$	$n + O(kd + \log k \log(1/\gamma))$	This work

Landscape of stewards

- Steward model captures derandomization constructions in literature

ε'	δ'	Randomness complexity	Reference
ε	$k\delta$	nk	Naïve
$O(\varepsilon)$	$2^k\delta$	n (works for $d = 1$ only)	This work
$O(\varepsilon d)$	$k\delta + \gamma$	$n + O(k \log(d+1) + \log k \log(1/\gamma))$	This work
$O(\varepsilon kd/\gamma)$	$k\delta + \gamma$	$n + O(k \log k + k \log d + k \log(1/\gamma))$	$\approx$ SZ '99
$O(\varepsilon)$	$k\delta + k/2^{n^{\Omega(1)}}$	$O(n^6 + kd)$	$\approx$ IZ '89
$O(\varepsilon)$	$k\delta + \gamma$	$n + O(kd + \log k \log(1/\gamma))$	This work
Any	Any ≤ 0.2	$n + \Omega(k) - \log(\delta'/\delta)$	This work

Open questions

- Optimal randomness complexity when d is large?

Open questions

- Optimal randomness complexity when d is large?
- Simultaneously achieve error $\varepsilon' \leq O(\varepsilon)$ and randomness complexity $n + O(k \log(d+1))$?

Open questions

- Optimal randomness complexity when d is large?
- Simultaneously achieve error $\varepsilon' \leq O(\varepsilon)$ and randomness complexity $n + O(k \log(d + 1))$?

- Thanks! Questions?
- This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1610403.