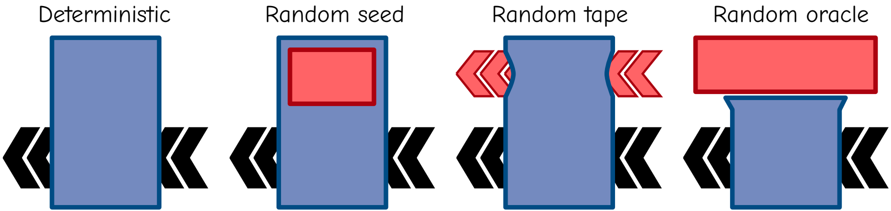
Finding Missing Items Requires Strong Form of Randomness

A. Chakrabarti[1] **M. Stoeckl**

[1]Department of Computer Science, Dartmouth College[1]

Computational Complexity Conference, 2024



[1]This work was supported in part by the National Science Foundation under award 2006589.

Outline

Problem and models

- Missing Item Finding
- Adversarial setting for streaming algorithms
- Types of randomness for streaming algorithms

Our Results/Contribution

- Separations
- Random tape algorithm
- Random tape lower bound
- Pseudo-deterministic lower bound

Open problems

Goal

For streaming algorithms in the adversarial setting, are there significant separations in space complexity for the *different ways randomness can be used?*

Yes: Missing Item Finding

Goal

For streaming algorithms in the adversarial setting, are there significant separations in space complexity for the *different ways randomness can be used?*

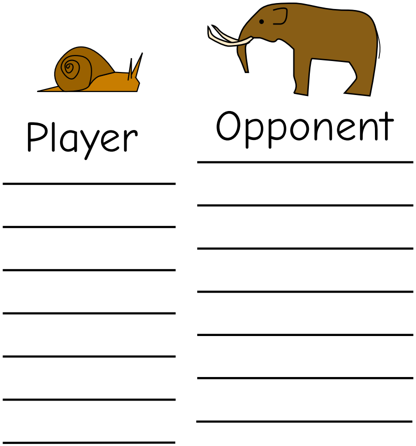
Yes: Missing Item Finding

Outline

The Missing Item Finding Problem

Definition 1.
$\mathrm{MIF}(n, \ell)$ for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not



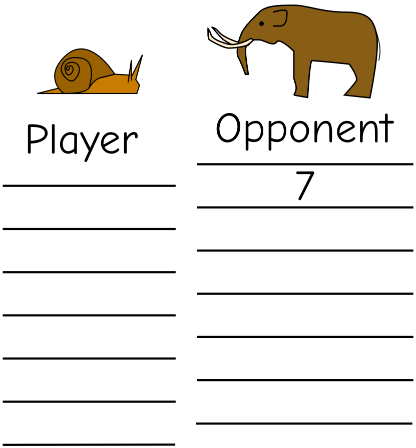
Example 2 (On right).
$MIF(n = 10, \ell = 6)$

Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.
$\mathsf{MIF}(n, \ell)$ for $1 \le \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not



Example 2 (On right).
$MIF(n = 10, \ell = 6)$

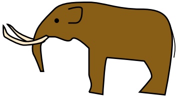
Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.

MIF(n, ℓ) for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not



Player

8

Opponent

7

Example 2 (On right).

$MIF(n = 10, \ell = 6)$

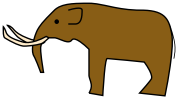
Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.
MIF(n, ℓ) for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \dots a_i\}$
- Player is memory limited; opponent is not

Example 2 (On right).
$MIF(n = 10, \ell = 6)$



Player	Opponent
	7
8	8

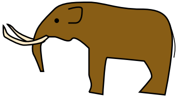
Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.
MIF(n, ℓ) for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \dots a_i\}$
- Player is memory limited; opponent is not

Example 2 (On right).
$MIF\,(n = 10, \ell = 6)$



Player	Opponent
8	7
4	8

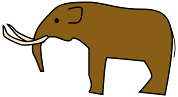
Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.
MIF(n, ℓ) for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \dots a_i\}$
- Player is memory limited; opponent is not

Example 2 (On right).
$MIF(n = 10, \ell = 6)$

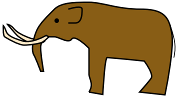
Player	Opponent
8	7
4	8
	9

Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.
$\mathrm{MIF}(n, \ell)$ for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not



Player	Opponent
8	7
4	8
4	9

Example 2 (On right).
$MIF(n = 10, \ell = 6)$

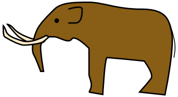
Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.
$\mathrm{MIF}(n, \ell)$ for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not

Example 2 (On right).
$MIF(n = 10, \ell = 6)$

Player	Opponent
	7
8	8
4	9
4	4

Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

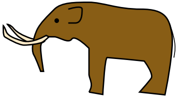
Definition 1.

MIF(n, ℓ) for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not

Example 2 (On right).

$MIF(n = 10, \ell = 6)$



Player	Opponent
8	7
4	8
4	9
3	4

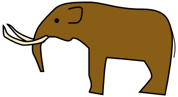
Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.
$\mathrm{MIF}(n, \ell)$ for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not

Example 2 (On right).
$MIF\,(n = 10, \ell = 6)$

Player	Opponent
	7
8	8
4	9
4	4
3	6

Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

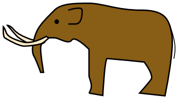
Definition 1.

MIF(n, ℓ) for $1 \le \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not

Example 2 (On right).

$MIF\,(n = 10, \ell = 6)$



Player	Opponent
8	7
4	8
4	9
3	4
5	6

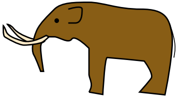
Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.
MIF(n, ℓ) for $1 \le \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not

Example 2 (On right).
$MIF(n = 10, \ell = 6)$



Player	Opponent
8	7
4	8
4	9
3	4
5	6
	5

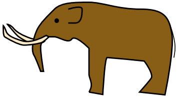
Two player game $\equiv$ Streaming algorithm with adaptive adversary

The Missing Item Finding Problem

Definition 1.
MIF(n, ℓ) for $1 \leq \ell < n$

- ℓ-step game:
  - Receive: $a_i \in [n]$
  - Output: $o_i \in [n] \setminus \{a_1, \ldots a_i\}$
- Player is memory limited; opponent is not

Example 2 (On right).
$MIF(n = 10, \ell = 6)$



Player	Opponent
	7
8	8
4	9
4	4
3	6
5	5
1	

Two player game $\equiv$ Streaming algorithm with adaptive adversary

Outline

Problem and models

- Missing Item Finding
- Adversarial setting for streaming algorithms
- Types of randomness for streaming algorithms

Our Results/Contribution

- Separations
- Random tape algorithm
- Random tape lower bound
- Pseudo-deterministic lower bound

Open problems

Streaming algorithms

- **Streaming Algorithm:**
  - limited memory
  - processes sequence of elements one by one

Algorithm for fixed MIF(n, ℓ) input streams with $\ell \ll n$

```
S ← random subset of [n] of size O(1)
for e from input stream
    if e ∈ S:
        remove e from S
    report: arbitrary element of S
```

Performance requirements

- This talk: state machine view (ignores description/computational cost).
- Tracking error: algorithm makes output after every input, correct iff all outputs are
- Algorithm has cost s if worst-case memory usage is s bits
- An algorithm has error $\leq \delta$ in the:
  - **static setting**: if it has error probability $\leq \delta$ for any input stream
  - **adversarial setting**:[2] if it has error probability $\leq \delta$ for *any adaptive adversary*

static setting $\rightarrow$ **classic algorithm** ; adversarial setting $\rightarrow$ **robust algorithm**

---

[2] *Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In Proc. 39th ACM Symposium on Principles of Database Systems, pages 63–80, 2020*

Performance requirements

- This talk: state machine view (ignores description/computational cost).
- Tracking error: algorithm makes output after every input, correct iff all outputs are
- Algorithm has cost s if worst-case memory usage is s bits
- An algorithm has error $\leq \delta$ in the:
  - **static setting**: if it has error probability $\leq \delta$ for any input stream
  - **adversarial setting**:[2] if it has error probability $\leq \delta$ for *any adaptive adversary*

static setting $\rightarrow$ **classic algorithm** ; adversarial setting $\rightarrow$ **robust algorithm**

[2] *Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In Proc. 39th ACM Symposium on Principles of Database Systems, pages 63–80, 2020*

Performance requirements

- This talk: state machine view (ignores description/computational cost).
- Tracking error: algorithm makes output after every input, correct iff all outputs are
- Algorithm has cost s if worst-case memory usage is s bits
- An algorithm has error $\leq \delta$ in the:
  - **static setting**: if it has error probability $\leq \delta$ for any input stream
  - **adversarial setting**:[2] if it has error probability $\leq \delta$ for *any adaptive adversary*

static setting $\to$ **classic algorithm** ; adversarial setting $\to$ **robust algorithm**

[2] *Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In Proc. 39th ACM Symposium on Principles of Database Systems, pages 63–80, 2020*

Performance requirements

- This talk: state machine view (ignores description/computational cost).
- Tracking error: algorithm makes output after every input, correct iff all outputs are
- Algorithm has cost s if worst-case memory usage is s bits
- An algorithm has error $\leq \delta$ in the:
  - **static setting**: if it has error probability $\leq \delta$ for any input stream
  - **adversarial setting**:[2] if it has error probability $\leq \delta$ for *any adaptive adversary*

static setting $\rightarrow$ **classic algorithm** ; adversarial setting $\rightarrow$ **robust algorithm**

[2] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. *A framework for adversarially robust streaming algorithms.* In Proc. 39th ACM Symposium on Principles of Database Systems, pages 63–80, 2020

Performance requirements

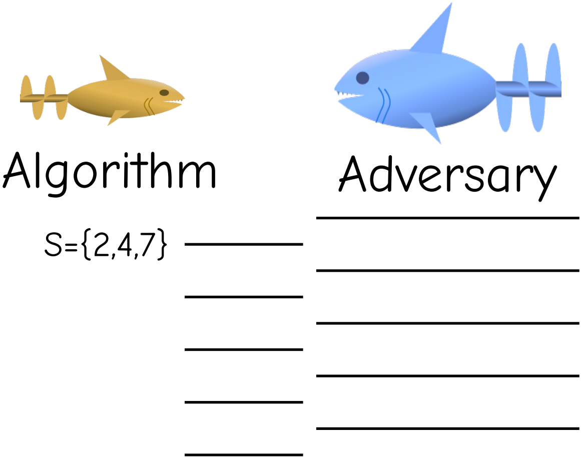
- This talk: state machine view (ignores description/computational cost).
- Tracking error: algorithm makes output after every input, correct iff all outputs are
- Algorithm has cost s if worst-case memory usage is s bits
- An algorithm has error $\leq \delta$ in the:
  - **static setting**: if it has error probability $\leq \delta$ for any input stream
  - **adversarial setting**:[2] if it has error probability $\leq \delta$ for *any adaptive adversary*

static setting $\rightarrow$ **classic algorithm** ; adversarial setting $\rightarrow$ **robust algorithm**

[2] *Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In Proc. 39th ACM Symposium on Principles of Database Systems, pages 63–80, 2020*

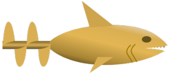
The adaptive adversary

- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness

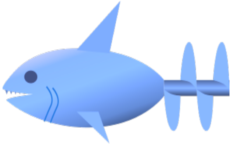


The adaptive adversary

- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness



Algorithm

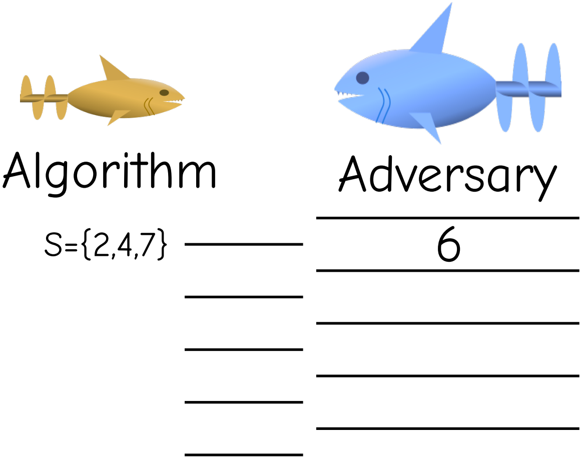


Adversary

S={2,4,7}

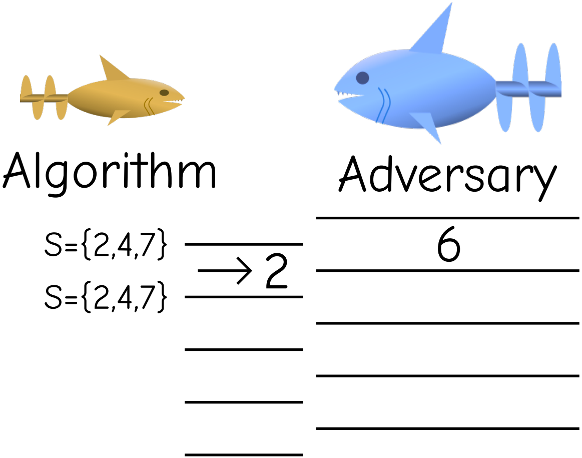
The adaptive adversary

- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness



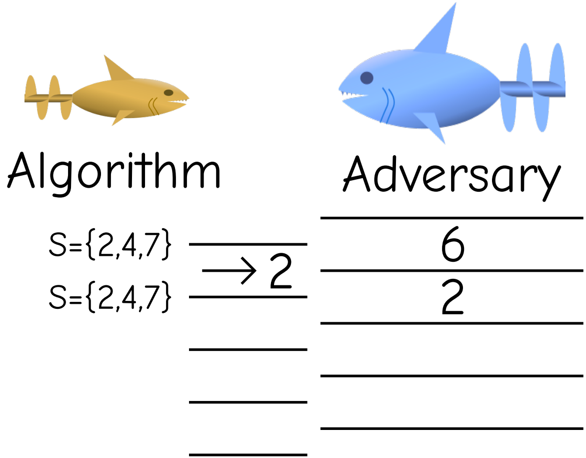
The adaptive adversary

- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness



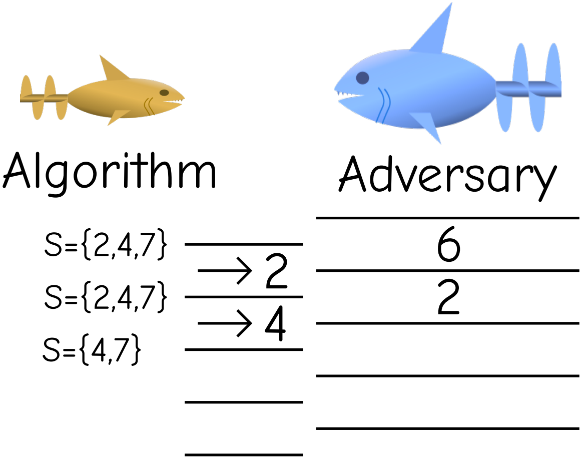
The adaptive adversary

- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness



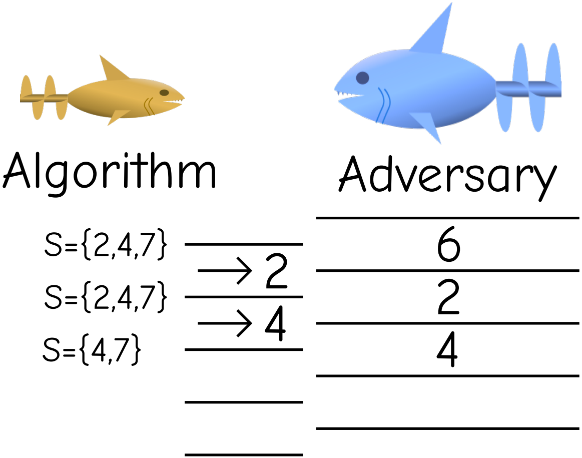
The adaptive adversary

- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness



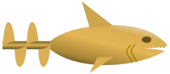
The adaptive adversary

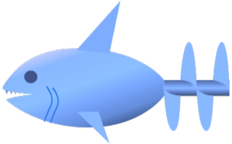
- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness



The adaptive adversary

- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness

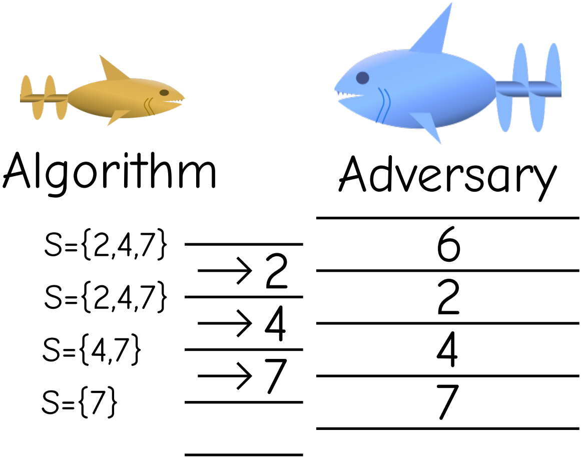




Algorithm		Adversary
S={2,4,7}		6
	→ 2	
S={2,4,7}		2
	→ 4	
S={4,7}		4
	→ 7	
S={7}		

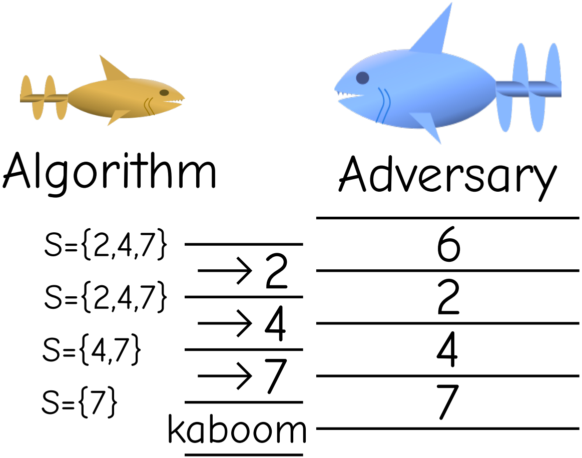
The adaptive adversary

- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness



The adaptive adversary

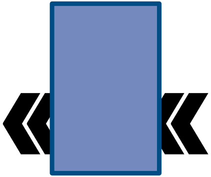
- Can choose next input element as function of preceding outputs
- This correlates input with private algorithm randomness



Outline

Four natural categories of streaming algorithms, by randomness type

Deterministic



No randomness

- Exact counter
- Sparse recovery
- Greedy matching

Random seed

Initial state random

- Linear sketch with random hash function
- Rabin fingerprint
- Example MIF algo

Random tape

Transitions random

- Reservoir sampling
- Morris Counter
- Noise / Differential privacy

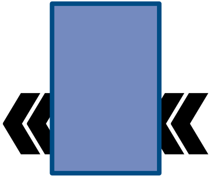
Random oracle

Free access to long, persistent, random string

- Linear sketch with i.i.d. entries

Four natural categories of streaming algorithms, by randomness type

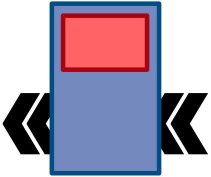
Deterministic



No randomness

- Exact counter
- Sparse recovery
- Greedy matching

Random seed



Initial state random

- Linear sketch with random hash function
- Rabin fingerprint
- Example MIF algo

Random tape

Transitions random

- Reservoir sampling
- Morris Counter
- Noise / Differential privacy

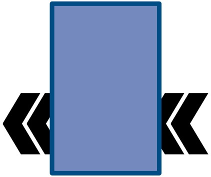
Random oracle

Free access to long, persistent, random string

- Linear sketch with i.i.d. entries

Four natural categories of streaming algorithms, by randomness type

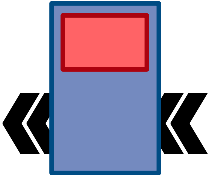
Deterministic



No randomness

- Exact counter
- Sparse recovery
- Greedy matching

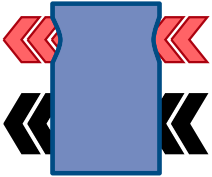
Random seed



Initial state random

- Linear sketch with random hash function
- Rabin fingerprint
- Example MIF algo

Random tape



Transitions random

- Reservoir sampling
- Morris Counter
- Noise / Differential privacy

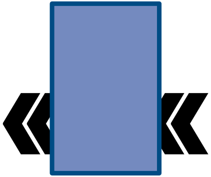
Random oracle

Free access to long, persistent, random string

- Linear sketch with i.i.d. entries

Four natural categories of streaming algorithms, by randomness type

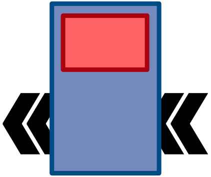
Deterministic



No randomness

- Exact counter
- Sparse recovery
- Greedy matching

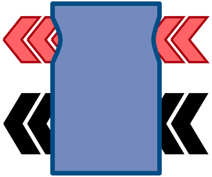
Random seed



Initial state random

- Linear sketch with random hash function
- Rabin fingerprint
- Example MIF algo

Random tape



Transitions random

- Reservoir sampling
- Morris Counter
- Noise / Differential privacy

Random oracle

Free access to long, persistent, random string

- Linear sketch with i.i.d. entries

Randomness type (almost) does not matter in static setting or with bounded adversaries

Emulate random oracle or random tape algorithm using random seed

Newman's theorem[a]

- Immediate corollary: any ϵ-error random oracle streaming algorithm with Q possible inputs has random seed emulation with $\epsilon(1+\delta)$ error and $+O\left(\log \frac{\log Q}{\epsilon\delta}\right)$ bits of space
- Non-constructive
- # adversaries $= \exp(\#$ streams$)$

[a] Ilan Newman. Private vs. common random bits in communication complexity. *Inform. Process. Lett.*, 39(2):67–71, 1991

Pseudo-random generators:

- If one-way functions exist and adversary is poly-time, ...
- If adversary has less memory than algorithm ...
  - Nisan's PRG[a]

[a] Noam Nisan. Pseudorandom generators for space-bounded computation. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 204–212, 1990

Randomness type (almost) does not matter in static setting or with bounded adversaries

Emulate random oracle or random tape algorithm using random seed

Newman's theorem[a]

- Immediate corollary: any ϵ-error random oracle streaming algorithm with Q possible inputs has random seed emulation with $\epsilon(1+\delta)$ error and $+O\left(\log \frac{\log Q}{\epsilon\delta}\right)$ bits of space
- Non-constructive
- # adversaries $= \exp(\#$ streams$)$

[a] Ilan Newman. Private vs. common random bits in communication complexity. *Inform. Process. Lett.*, 39(2):67–71, 1991

Pseudo-random generators:

- If one-way functions exist and adversary is poly-time, ...
- If adversary has less memory than algorithm ...
  - Nisan's PRG[a]

[a] Noam Nisan. Pseudorandom generators for space-bounded computation. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 204–212, 1990

Randomness type (almost) does not matter in static setting or with bounded adversaries

Emulate random oracle or random tape algorithm using random seed

Newman's theorem[a]

- Immediate corollary: any ϵ-error random oracle streaming algorithm with Q possible inputs has random seed emulation with $\epsilon(1+\delta)$ error and $+O\left(\log \frac{\log Q}{\epsilon\delta}\right)$ bits of space
- Non-constructive
- # adversaries $= \exp(\#$ streams$)$

[a]Ilan Newman. Private vs. common random bits in communication complexity. *Inform. Process. Lett.*, 39(2):67–71, 1991

Pseudo-random generators:

- If one-way functions exist and adversary is poly-time, ...
- If adversary has less memory than algorithm ...
  - Nisan's PRG[a]

[a]Noam Nisan. Pseudorandom generators for space-bounded computation. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 204–212, 1990

Outline

Main Result

For streaming algorithms in the adversarial setting, are there significant separations in space complexity for different ways randomness can be used?

MIF (n, ℓ) space, adversarial setting

	$\ell = 2^{\sqrt{\log n}}$	$\ell = \sqrt{n}$
Random oracle	$\tilde{\Theta}(1)$	$\tilde{\Theta}(1)$
Random tape	$\tilde{\Theta}(1)$	$\tilde{\Omega}(\ell^{1/4})$
Random seed	$\tilde{\Theta}(\sqrt{\ell})$	$\tilde{\Theta}(\sqrt{\ell})$
Deterministic	$\tilde{\Theta}(\ell)$	$\tilde{\Theta}(\ell)$

(Hiding polylog(n, ℓ) factors; at error $\delta = \frac{1}{n^2}$)

Yes: for RT/RO and RS/RT

Main Result

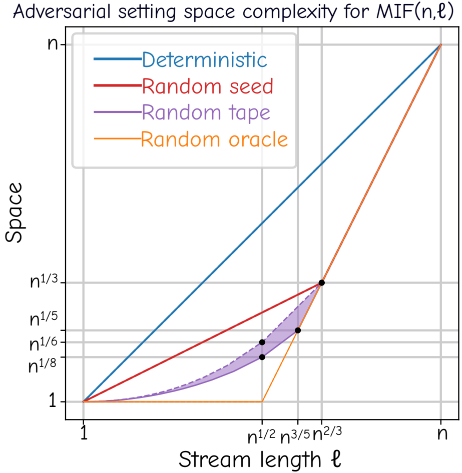
For streaming algorithms in the adversarial setting, are there significant separations in space complexity for different ways randomness can be used?

MIF (n, ℓ) space, adversarial setting

	$\ell = 2^{\sqrt{\log n}}$	$\ell = \sqrt{n}$
Random oracle	$\tilde{\Theta}(1)$	$\tilde{\Theta}(1)$
Random tape	$\tilde{\Theta}(1)$	$\tilde{\Omega}(\ell^{1/4})$
Random seed	$\tilde{\Theta}\left(\sqrt{\ell}\right)$	$\tilde{\Theta}\left(\sqrt{\ell}\right)$
Deterministic	$\tilde{\Theta}(\ell)$	$\tilde{\Theta}(\ell)$

(Hiding polylog(n, ℓ) factors; at error $\delta = \frac{1}{n^2}$)

Yes: for RT/RO and RS/RT



Specific results of this paper

1. Lower bound for random tape in adversarial setting
2. Upper bound for random tape in adversarial setting
3. Lower bound for *pseudo-deterministic* algorithms
4. Corollary via older work[a]: $\implies$ lower bound for random *seed* in adversarial setting

[a] Manuel Stoeckl. Streaming algorithms for the missing item finding problem. In *Proc. 34th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 793–818, 2023. Full version at arXiv:2211.05170v1

Known results (this + Stoeckl 2023, 2024)

Setting	Type	Space
Static	Seed	$\tilde{\Theta}(1)$
Adversarial	Oracle	$\tilde{\Theta}(\ell^2/n + 1)$
Adversarial	Tape	$\Omega\left(\ell^{\frac{15}{32}\log_n \ell}\right)$ $\tilde{O}\left(\ell^{\log_n \ell}\right)$
Adversarial	Seed	$\tilde{\Theta}\left(\ell^2/n + \sqrt{\ell}\right)$
Pseudo-det.	Oracle	$\tilde{\Theta}(\ell)$
Any	Det.	$\tilde{\Theta}(\ell)$

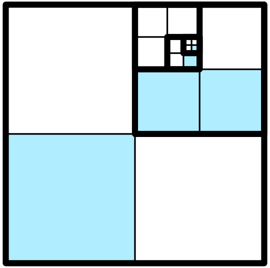
(Hiding polylog(n, ℓ) factors; at error $\delta = \frac{1}{n^2}$)

Other work: Magen 2024; Tarui 2007

Outline

Random tape algorithm (in adversarial setting): recursive structure

- Design: split universe $[n]$ into k blocks
- Run random-subset algorithm to choose a safe block
- Inside chosen safe block: run this algorithm on domain $[n/k]$



Init()

```
    Split [n] into B_1,...,B_k
    S ← random subset of [k]
    c ← S.pop()
    A ← recursive instance on B_c
```

Update(e)

```
    if e ∈ B_i for any i in S:
        remove i from S
    if e ∈ B_c:
        A.Update(e)
        if A is done:
            c ← S.pop()?
            A ← recursive instance on B_c
    report: A's output
```

Outline

Random tape lower bound (in adversarial setting): recursive structure

Lemma 3 (Stoeckl 2023).
Robust algorithms for MIF (n, ℓ) with $\leq \frac{3}{4}$ error probability require $\Omega\left(\ell^2/n\right)$ space

Lemma 4.
Lower bound for a z-bit robust random tape algorithm for MIF (n, ℓ) depends on the lower bound for MIF (w, t) with

$$w = \Theta\left(\frac{zn}{\ell}\right) \qquad t = \Theta\left(\frac{\ell}{z}\right)$$

Theorem 5.
Robust random tape algorithms for MIF (n, ℓ) require space:

$$\Omega\left(\max_k \left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}\right) = \Omega\left(\ell^{\frac{15}{32}\log_n \ell}\right)$$

Reduction step: searching for information must stop

1. Adversary sends $\ell/2$ random elements
   - Let ρ be resulting algorithm state
   - Not enough space to store all random elements: algorithm must overestimate, and only considers elements in a set H_ρ to be *safe*
   - Typically $|H_\rho| = O\left(\frac{zn}{\ell}\right)$
2. Adversary tries to identify H_ρ, in $\Theta(z)$ epochs
   - Have a possible set H_σ for each possible state σ; making an output outside H_σ is risky if $\sigma = \rho$
   - Each epoch, either:
     (a) There exists a "sub-adversary" for next $\Theta\left(\frac{\ell}{z}\right)$ steps which likely rules out half the remaining candidate H_σ values. If so, run it!
     (b) There exists a set W of size $O\left(\frac{zn}{\ell}\right)$ which probably contains *all* the next $\Theta\left(\frac{\ell}{z}\right)$ algorithm outputs, no matter what
   - Case (a) is unlikely to happen $\Theta(z)$ times – might end up ruling out H_ρ itself
   - Case (b): DONE – algorithm solves $MIF\left(O\left(\frac{zn}{\ell}\right), \Theta\left(\frac{\ell}{z}\right)\right)$ with inputs in W

Reduction step: searching for information must stop

1. Adversary sends $\ell/2$ random elements
   - Let ρ be resulting algorithm state
   - Not enough space to store all random elements: algorithm must overestimate, and only considers elements in a set H_ρ to be *safe*
   - Typically $|H_\rho| = O\left(\frac{zn}{\ell}\right)$
2. Adversary tries to identify H_ρ, in $\Theta(z)$ epochs
   - Have a possible set H_σ for each possible state σ; making an output outside H_σ is risky if $\sigma = \rho$
   - Each epoch, either:
     (a) There exists a "sub-adversary" for next $\Theta\left(\frac{\ell}{z}\right)$ steps which likely rules out half the remaining candidate H_σ values. If so, run it!
     (b) There exists a set W of size $O\left(\frac{zn}{\ell}\right)$ which probably contains *all* the next $\Theta\left(\frac{\ell}{z}\right)$ algorithm outputs, no matter what
   - Case (a) is unlikely to happen $\Theta(z)$ times – might end up ruling out H_ρ itself
   - Case (b): DONE – algorithm solves $MIF\left(O\left(\frac{zn}{\ell}\right), \Theta\left(\frac{\ell}{z}\right)\right)$ with inputs in W

Outline

- Problem and models
  - Missing Item Finding
  - Adversarial setting for streaming algorithms
  - Types of randomness for streaming algorithms
- Our Results/Contribution
  - Separations
  - Random tape algorithm
  - Random tape lower bound
  - Pseudo-deterministic lower bound
- Open problems

What is pseudo-determinism?

- Randomized algorithm $\mathcal{A}$ that behaves like a deterministic one
  - There exists **canonical output function** $f_{\mathcal{A}}$ from inputs to outputs so that $\Pr\left[\mathcal{A}(x) = f_{\mathcal{A}}(x)\right] \geq 1 - \epsilon$ for all possible inputs x
- Pseudo-deterministic streaming algorithms:[3]
  - If correctness relation is a function, correct algorithms are pseudo-deterministic
  - Automatically work in the adversarial setting
  - Newman's theorem can apply

[3] For a paper introducing pseudo-determinism to streaming, see: Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. Pseudo-Deterministic Streaming. In *Proc. 20th Conference on Innovations in Theoretical Computer Science*, volume 151, 79:1–79:25, 2020

What is pseudo-determinism?

- Randomized algorithm $\mathcal{A}$ that behaves like a deterministic one
  - There exists **canonical output function** $f_{\mathcal{A}}$ from inputs to outputs so that $\Pr\left[\mathcal{A}(x) = f_{\mathcal{A}}(x)\right] \geq 1 - \epsilon$ for all possible inputs x
- Pseudo-deterministic streaming algorithms:[3]
  - If correctness relation is a function, correct algorithms are pseudo-deterministic
  - Automatically work in the adversarial setting
  - Newman's theorem can apply

[3] For a paper introducing pseudo-determinism to streaming, see: Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. Pseudo-Deterministic Streaming. *In Proc. 20th Conference on Innovations in Theoretical Computer Science*, volume 151, 79:1–79:25, 2020

Pseudo-deterministic and random seed lower bounds

Theorem 6.

Pseudo-deterministic random-oracle algorithms for MIF (n, ℓ) with error $\delta = 1/poly(n)$ and $\ell = \Omega(\log n)$ require space

$$\Omega\left(\frac{\ell}{(\log n)^2}\right)$$

Theorem 7 (Stoeckl 2024).

A random seed streaming algorithm for adversarial setting with z bits of state processing a stream of length ℓ can be made to probably "behave pseudo-deterministically" for some contiguous stretch of $\Theta(\ell/z)$ inputs.

Corollary 8.

Random seed, adversarial setting, $\leq 1/6$ error, MIF (n, ℓ) algorithms require space[4]

$$\Omega\left(\frac{\ell^2}{n} + \sqrt{\frac{\ell}{(\log n)^3}}\right)$$

[4] The ℓ^2/n term comes from Lemma 3

Open problems

- Mirror Game: like MIF, but a) neither player can repeat numbers b) $n = 2\ell$ c) player starts[5]
  - Unknown: do space-efficient algorithms need a random oracle?
- Can we separate random *seed* and *tape* for adversarial setting turnstile L_0 estimation algorithms? Pseudo-deterministic gap hamming communication complexity still open.[6]

[5] Sumegha Garg and Jon Schneider. The Space Complexity of Mirror Games. In *Proc. 10th Conference on Innovations in Theoretical Computer Science*, 36:1–36:14, 2018, Feige 2019; Magen and Naor 2022; Menuhin and Naor 2022

[6] *Some progress:* Dmytro Gavinsky. Unambiguous parity-query complexity. *arXiv preprint arXiv:2401.11274*, 2024

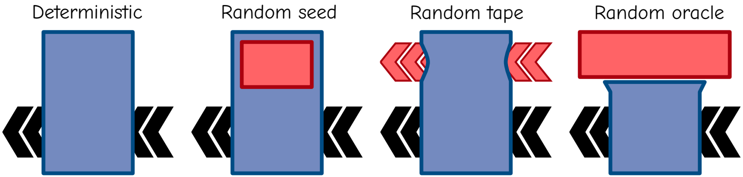
Open problems

- Mirror Game: like MIF, but a) neither player can repeat numbers b) $n = 2\ell$ c) player starts[5]
  - Unknown: do space-efficient algorithms need a random oracle?
- Can we separate random *seed* and *tape* for adversarial setting turnstile L_0 estimation algorithms? Pseudo-deterministic gap hamming communication complexity still open.[6]

[5] Sumegha Garg and Jon Schneider. The Space Complexity of Mirror Games. *In Proc. 10th Conference on Innovations in Theoretical Computer Science*, 36:1–36:14, 2018, Feige 2019; Magen and Naor 2022; Menuhin and Naor 2022

[6] *Some progress:* Dmytro Gavinsky. Unambiguous parity-query complexity. *arXiv preprint arXiv:2401.11274*, 2024

Conclusion



- Unlike in the static setting, the example of Missing Item Finding shows that space-efficient streaming algorithms in the adversarial setting may require a random tape or random oracle.

- Lower bound methods:
  - Random tape: Recursive structure of MIF algorithms + adversary iteratively searching for information on past states + a useful property when search cannot progress
  - Random seed: use semi-generic reduction to pseudo-deterministic
  - Pseudo-deterministic: generalize deterministic proof + alternate establishing canonical and actual algorithm properties

Bibliography I

Miklós Ajtai, Vladimir Braverman, T.S. Jayram, Sandeep Silwal, Alec Sun, David P. Woodruff, and Samson Zhou. The white-box adversarial data stream model. In *Proc. 41st ACM Symposium on Principles of Database Systems*, pages 15–27, 2022.

Sepehr Assadi, Andrew Chen, and Glenn Sun. Deterministic graph coloring in the streaming model. In *Proc. 54th Annual ACM Symposium on the Theory of Computing*, pages 261–274, 2022.

Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. Adversarially robust streaming via dense-sparse trade-offs. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 214–227, 2022.

Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In *Proc. 39th ACM Symposium on Principles of Database Systems*, pages 63–80, 2020.

Bibliography II

Omri Ben-Eliezer and Eylon Yogev. The adversarial robustness of sampling. In *Proc. 39th ACM Symposium on Principles of Database Systems*, pages 49–62. ACM, 2020.

Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Adversarially robust coloring for graph streams. In *Proc. 13th Conference on Innovations in Theoretical Computer Science*, 37:1–37:23, 2022.

Uriel Feige. A randomized strategy in the mirror game. *arXiv preprint arXiv:1901.07809*, 2019.

Dmytro Gavinsky. Unambiguous parity-query complexity. *arXiv preprint arXiv:2401.11274*, 2024.

Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. Pseudo-Deterministic Streaming. In *Proc. 20th Conference on Innovations in Theoretical Computer Science*, volume 151, 79:1–79:25, 2020.

Bibliography III

Sumegha Garg and Jon Schneider. The Space Complexity of Mirror Games. In *Proc. 10th Conference on Innovations in Theoretical Computer Science*, 36:1–36:14, 2018.

Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. Separating adaptive streaming from oblivious streaming using the bounded storage model. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 94–121. Springer, 2021.

Bibliography IV

Roey Magen. Are we still missing an item? *arXiv preprint arXiv:2401.06547*, 2024.

Roey Magen and Moni Naor. Mirror games against an open book player. In *11th International Conference on Fun with Algorithms (FUN 2022)*, volume 226, 20:1–20:12, 2022.

Boaz Menuhin and Moni Naor. Keep that card in mind: card guessing with limited memory. In *Proc. 13th Conference on Innovations in Theoretical Computer Science*, 107:1–107:28, 2022.

Ilan Newman. Private vs. common random bits in communication complexity. *Inform. Process. Lett.*, 39(2):67–71, 1991.

Noam Nisan. Pseudorandom generators for space-bounded computation. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 204–212, 1990.

Bibliography V

Noam Nisan. On read once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107(1):135–144, 1993.

Manuel Stoeckl. Streaming algorithms for the missing item finding problem. In *Proc. 34th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 793–818, 2023. Full version at arXiv:2211.05170v1.

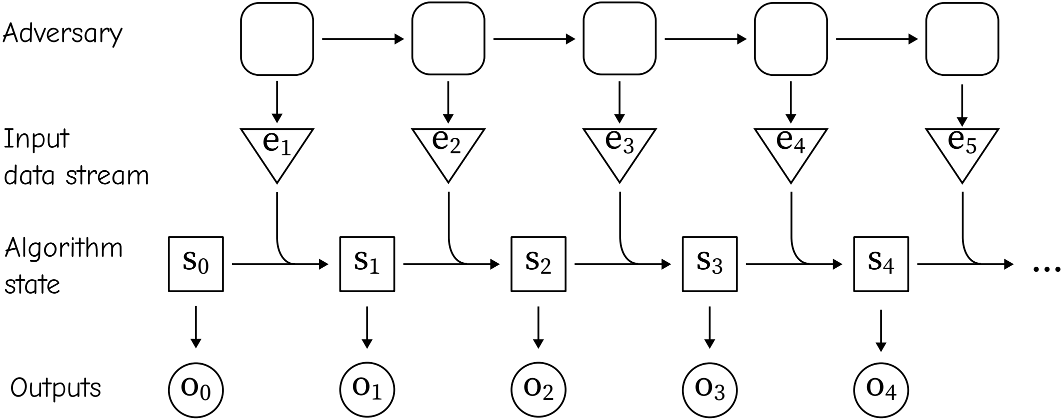
Manuel Stoeckl. *On adaptivity and randomness for streaming algorithms*. PhD thesis, Dartmouth College, 2024.

Jun Tarui. Finding a duplicate and a missing item in a stream. In *Proc. 4th International Conference on Theory and Applications of Models of Computation*, pages 128–135, 2007.

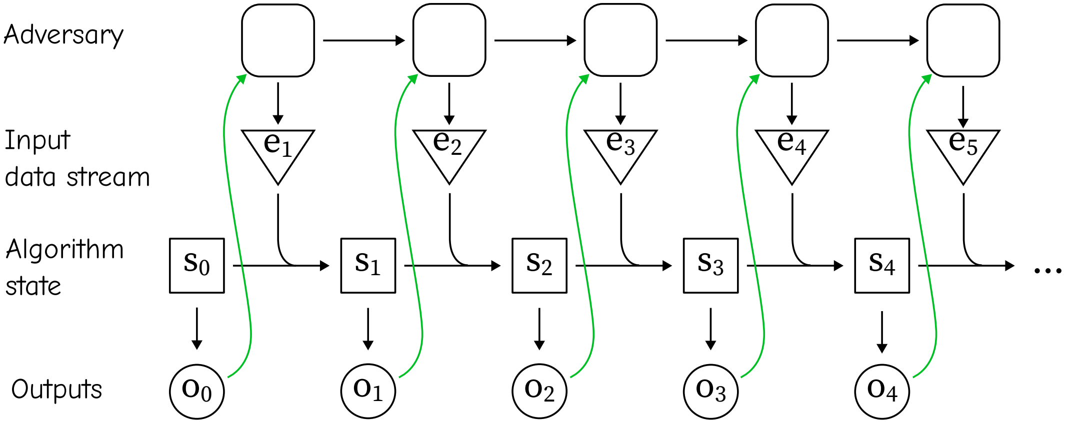
Bonus slides

1. State machine views (of randomness types, models.)
2. Random oracle algorithm explanation
3. Random-seed to pseudo-deterministic explanation
4. Full statements of main theorems
5. Reduction step for random tape lower bound
6. Simplified FindCommonOutputs
7. Related work

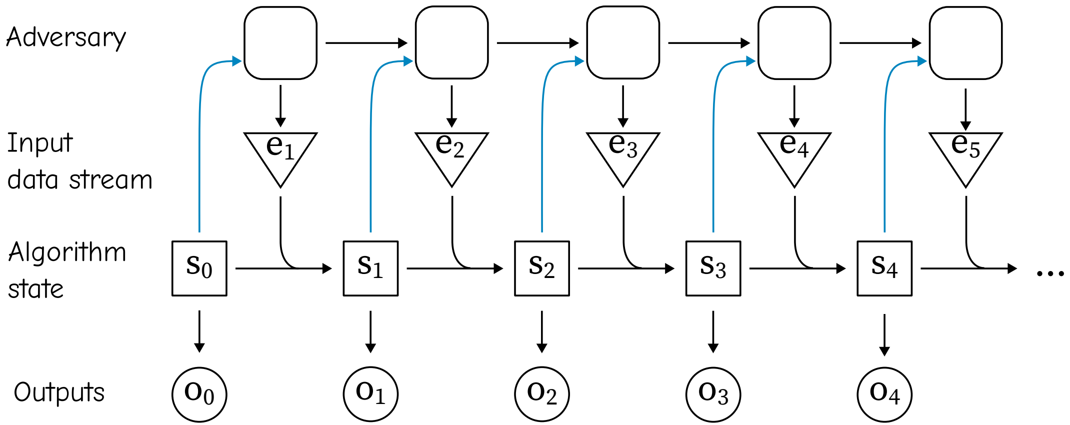
State machine perspective: static setting



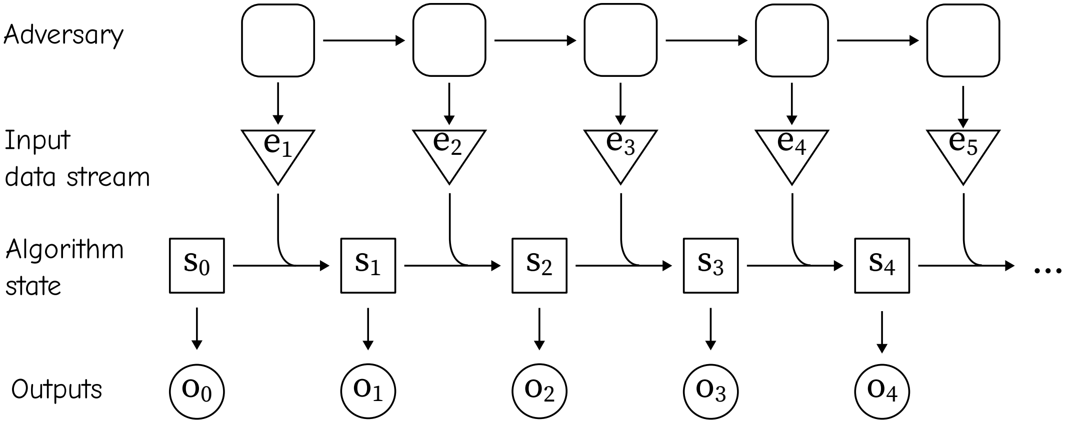
State machine perspective: adversarial setting



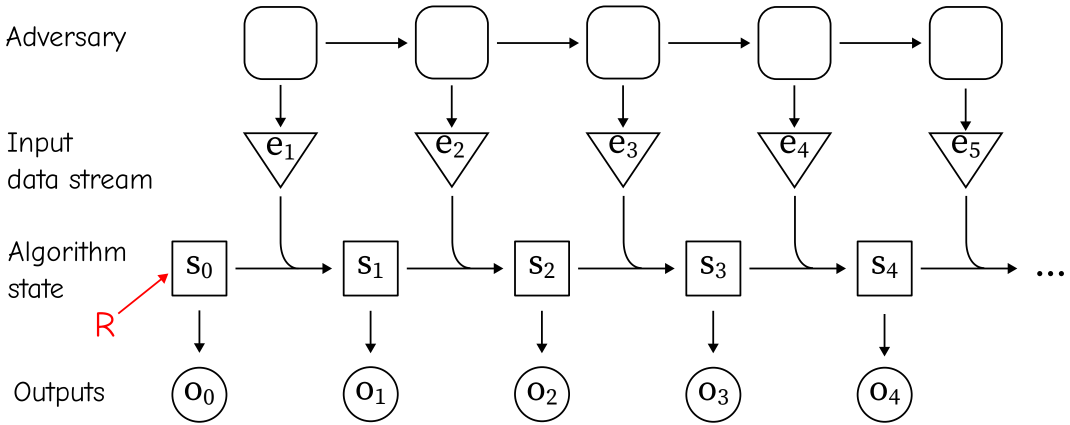
State machine perspective: white-box adversarial setting



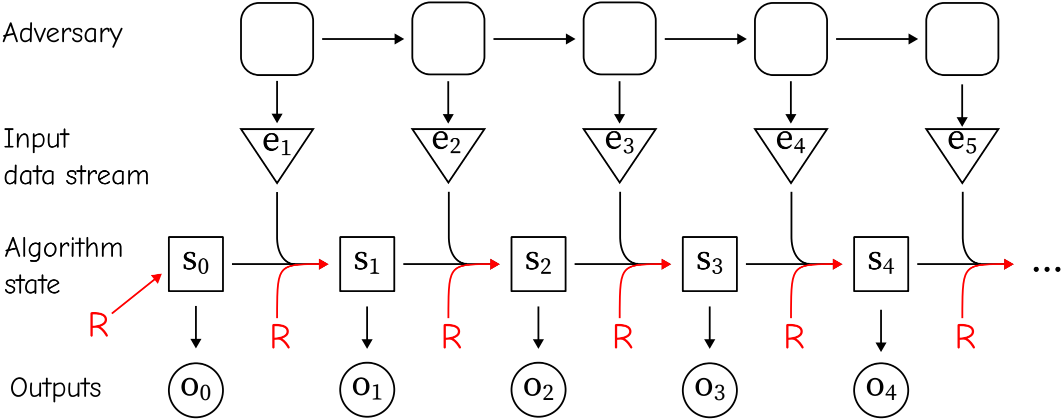
State machine perspective: deterministic



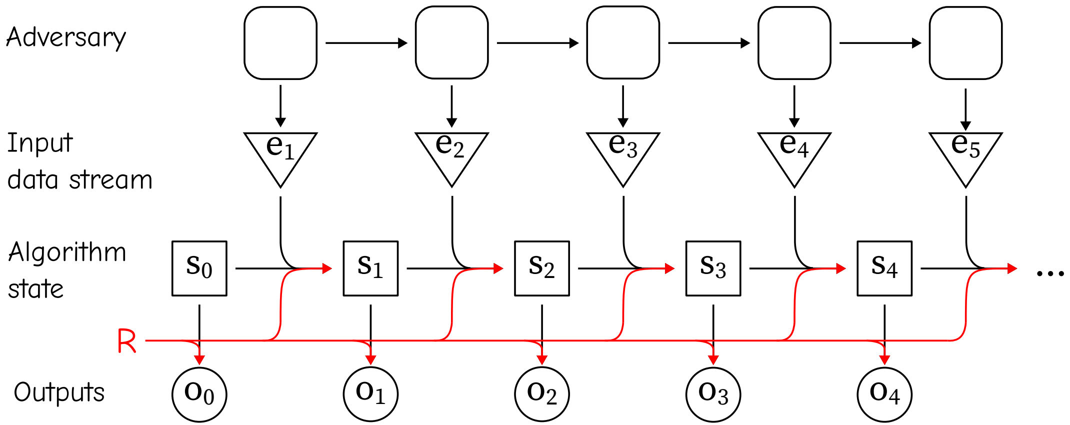
State machine perspective: random seed



State machine perspective: random tape

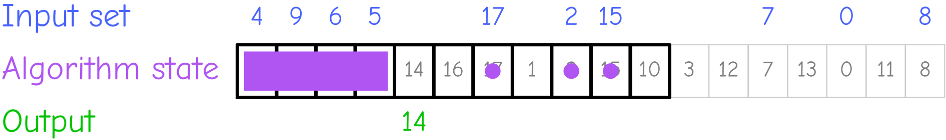


State machine perspective: random oracle



Random oracle algorithm

- In "random sample" approach, S can be drawn from oracle randomness
- If always outputting *least* available element of S, can efficiently encode removed elements as union of contiguous and sparse sets



Random seed to pseudo-deterministic

Consider adversary with $\Theta(z)$ epochs of length $t = \Theta(\ell/z)$.

For each epoch:

1. If $\exists$ subsequence x of length t for which, conditioned on history, algorithm output sequence has high entropy (≥ 0.5 bits, say):
   - Send x to algorithm. Next epoch.
2. Otherwise, for all possible x, conditional entropy of outputs is low (≤ 0.5 bits) which implies some particular output sequence occurs with probability $\geq \frac{2}{3}$.[7] This is pseudo-determinism.

Observe: entropy of random seed is limited by z, so case 1 can only occur $\geq 4z$ times, a constant fraction of the time.[8]

---

[7] Say all output sequences have $\leq \frac{2}{3}$ probability. Then there exists a subset S of possible outputs with net probability between $\frac{1}{3}$ and $\frac{2}{3}$; $H(X \in S) \geq -\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3} \geq 0.798$.

[8] $H(S) \geq H(O_1) + H(O_2|O_1) + \ldots) \geq \frac{1}{2} \cdot \frac{1}{2} \cdot 4z$; last step is hiding expansion into events via $H(X|Y) = \sum p(y) H(X|Y=y)$ and filtering by $\geq 4z$ type-1 steps.

Formal theorem statements

Theorem 9.
Random tape δ-error adversarially robust algorithms for MIF (n, ℓ), with $\delta \leq \frac{\ell}{2^7 n}$, require space:

$$\Omega\left(\max_{k \in \mathbb{N}} \left(\frac{\ell^{k+1}}{n}\right)^{\frac{2}{k^2+3k-2}}\right) = \Omega\left(\ell^{\frac{15}{32}\log_n \ell}\right)$$

Theorem 10.
There is a family of adversarially robust random tape algorithms, where for MIF (n, ℓ) the corresponding algorithm has $\leq \delta$ error and uses

$$O\left(\left\lceil \frac{(4\ell)^{\frac{2}{d-1}}}{(n/4)^{\frac{3}{d(d-1)}}} \right\rceil (\log \ell)^2 + \min\left(\ell, \log\frac{1}{\delta}\right)\log \ell\right)$$

bits of space, where $d = \max\left(2, \min\left(\lceil \log \ell \rceil, \left\lfloor 2\frac{\log n/4}{\log(16\ell)} \right\rfloor\right)\right)$. When $\delta = 1/poly(n)$ a weakened space bound is $O\left(\ell^{\log_n \ell}(\log \ell)^2 + \log \ell\right)$.

Formal theorem statements

Theorem 11.
Pseudo-deterministic δ-error random oracle algorithms for MIF (n, ℓ) require

$$\Omega\left(\min\left(\frac{\ell}{\log\frac{2n}{\ell}} + \sqrt{\ell}, \frac{\ell\log\frac{1}{2\delta}}{\left(\log\frac{2n}{\ell}\right)^2\log n} + \left(\ell\log\frac{1}{2\delta}\right)^{1/4}\right)\right)$$

bits of space when $\delta \leq \frac{1}{3}$. In particular, when $\delta = 1/poly(n)$ and $\ell = \Omega(\log n)$, this is

$$\Omega\left(\frac{\ell}{\left(\log\frac{2n}{\ell}\right)^2} + (\ell\log n)^{1/4}\right)$$

Theorem 12.
Adversarially robust random seed algorithms for MIF (n, ℓ) with error $\leq \frac{1}{6}$ require space:

$$\Omega\left(\frac{\ell^2}{n} + \sqrt{\frac{\ell}{(\log n)^3}} + \ell^{1/5}\right)$$

Reduction step: searching for information must stop

1. Adversary sends $\ell/2$ random elements
   - Let ρ be resulting algorithm state
   - Not enough space to store all random elements: algorithm must overestimate, and only considers elements in a set H_ρ to be *safe*
   - Typically $|H_\rho| = O\left(\frac{zn}{\ell}\right)$
2. Adversary tries to identify H_ρ, in $\Theta(z)$ epochs
   - Have a possible set H_σ for each possible state σ; making an output outside H_σ is risky if $\sigma = \rho$
   - Each epoch, either:
     (a) There exists a "sub-adversary" for next $\Theta\left(\frac{\ell}{z}\right)$ steps which likely rules out half the remaining candidate H_σ values. If so, run it!
     (b) There exists a set W of size $O\left(\frac{zn}{\ell}\right)$ which probably contains *all* the next $\Theta\left(\frac{\ell}{z}\right)$ algorithm outputs, no matter what
   - Case (a) is unlikely to happen $\Theta(z)$ times – might end up ruling out H_ρ itself
   - Case (b): DONE – algorithm solves $MIF\left(O\left(\frac{zn}{\ell}\right), \Theta\left(\frac{\ell}{z}\right)\right)$ with inputs in W

Reduction step: searching for information must stop

1. Adversary sends $\ell/2$ random elements
   - Let ρ be resulting algorithm state
   - Not enough space to store all random elements: algorithm must overestimate, and only considers elements in a set H_ρ to be *safe*
   - Typically $|H_\rho| = O\left(\frac{zn}{\ell}\right)$
2. Adversary tries to identify H_ρ, in $\Theta(z)$ epochs
   - Have a possible set H_σ for each possible state σ; making an output outside H_σ is risky if $\sigma = \rho$
   - Each epoch, either:
     (a) There exists a "sub-adversary" for next $\Theta\left(\frac{\ell}{z}\right)$ steps which likely rules out half the remaining candidate H_σ values. If so, run it!
     (b) There exists a set W of size $O\left(\frac{zn}{\ell}\right)$ which probably contains *all* the next $\Theta\left(\frac{\ell}{z}\right)$ algorithm outputs, no matter what
   - Case (a) is unlikely to happen $\Theta(z)$ times – might end up ruling out H_ρ itself
   - Case (b): DONE – algorithm solves *MIF* $\left(O\left(\frac{zn}{\ell}\right), \Theta\left(\frac{\ell}{z}\right)\right)$ with inputs in W

Very simplified proof sketch

Generalization of deterministic lower bound from Stoeckl 2023

- For any state σ, integer q, let $FCO(\sigma, q)$ be set of "most common outputs" after q more inputs, with size w_q
- Interpret partial input stream $x \in [n]^\star$ as a state of the "canonical protocol"; then $FCO(x, q)$ gives most common canonical outputs
- We can recursively define FCO and hence "common outputs" so that we can prove:
  - If σ is a random state resulting from input x, then w.h.p. $FCO(\sigma, q) = FCO(x, q)$
  - $FCO(x, q) \cap x = \emptyset$
  - $|FCO(x, q)| \approx 2^{q/z}$, where the algorithm uses z bits of state
    - Pseudo-determinism used here: output built from dependent evaluations
- Since $n \geq |FCO(\epsilon, \ell)| \approx 2^{\ell/z}$, it follows $z \gtrapprox \frac{\ell}{\log n}$

FindCommonOutputs

- B is input to output function implemented by algorithm or canonical; $C \in_R [1,2)^{d \times N}$, x is stream prefix, and epochs are $t_d + \ldots + t_1 = \ell$; x has length $t_d + \ldots + t_{k+1}$. S is set of possible canonical outputs. $FCO(\cdots, k)$ output size is w_k, with all $w_k \geq \frac{5}{4} w_{k-1}$.

FindCommonOutputs(B, C, x, k)[9]

if $k = 1$

return iteratively extracted w_1 distinct elements, or error

$Q \leftarrow FCO(B, C, x \circ \langle 1, \ldots, t_k \rangle, k-1)$

for each $j \in S$

$f_j \leftarrow \left| \left\{ y \in \binom{Q}{t_k} : j \in FCO(B, C, x \circ \mathrm{sorted}(y), k-1) \right\} \right|$

$\theta \leftarrow C_{k,h} w_{k-1} / 16 |S|$

$P \leftarrow \left\{ j \in S : f_j^{(h)} \geq \theta \binom{|Q|}{t_k} \right\}$

return first w_k elements of $Q \cup P$

[9]This includes a simplification not present in published work.

Related work

- Generic methods to convert static to adversarial setting: Ben-Eliezer, Jayaram, Woodruff, and Yogev 2020; Ben-Eliezer and Yogev 2020, and recent diff. privacy approaches (which use random-tape) Ben-Eliezer, Eden, and Onak 2022; Hassidim, Kaplan, Mansour, Matias, and Stemmer 2020
- Static vs. adversarial separations: Assadi, Chen, and G. Sun 2022; Chakrabarti, Ghosh, and Stoeckl 2022; Kaplan, Mansour, Nissim, and Stemmer 2021
- White-box adversaries: Ajtai, Braverman, Jayram, Silwal, A. Sun, Woodruff, and Zhou 2022

- Read-once vs read-multiple use of randomness: Nisan 1993[10]
- Other work on Missing Item Finding and variants: Chakrabarti, Ghosh, and Stoeckl 2022; Magen 2024; Stoeckl 2023, 2024; Tarui 2007

[10] Noam Nisan. On read once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107(1):135–144, 1993