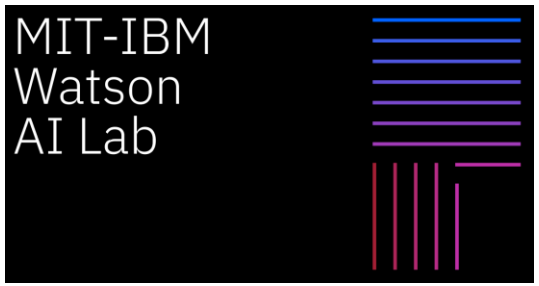
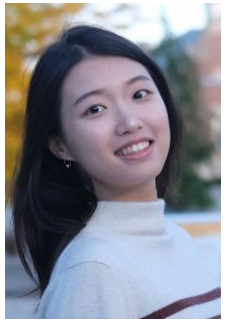
Information-Theoretic Driven Watermarking of Large-Language Models

Dor Tsur, October 17th 2025.

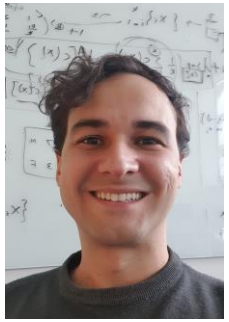


Collaborators

Carol Long
Harvard University



Claudio Mayrink Verdun
Harvard University



Hsiang Hsu
JP Morgan & Chase

Chun-Fu (Richard) Chen
JP Morgan & Chase

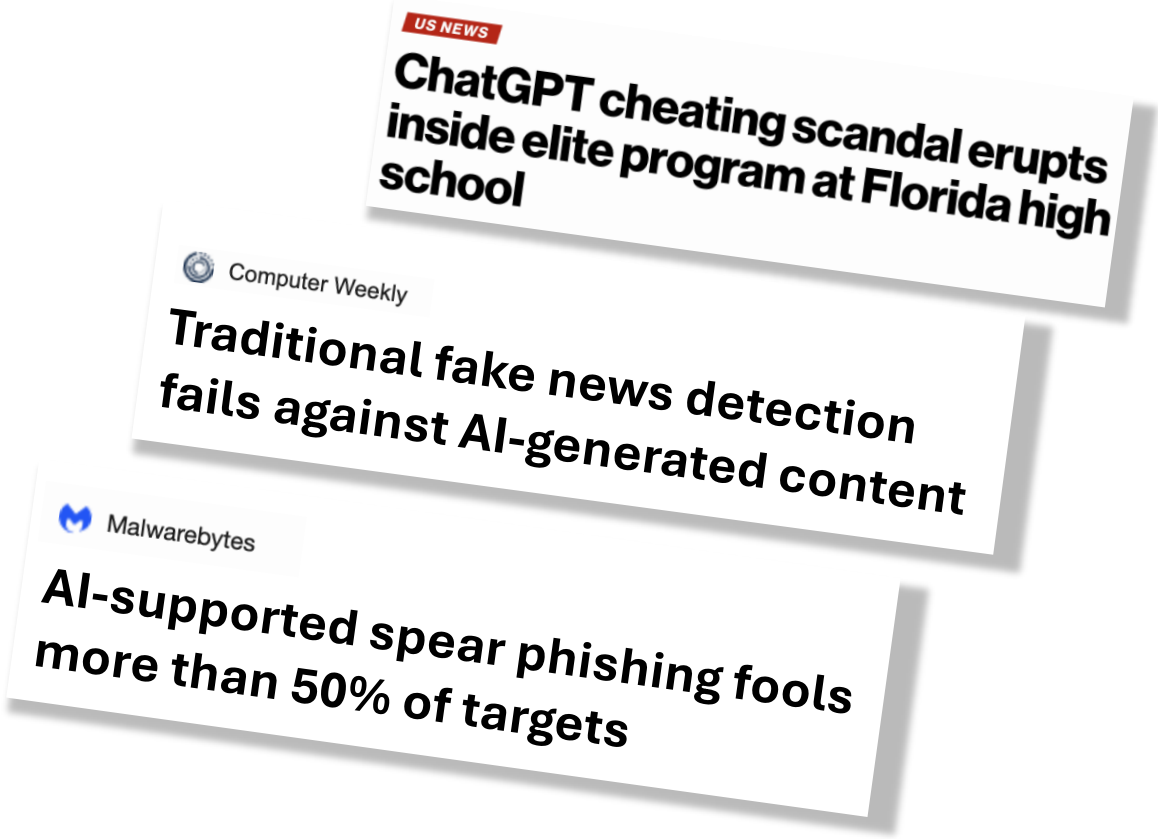
Haim H. Permuter
Ben-Gurion University

Sajani Vithana
Harvard University

Flavio P. Calmon
Harvard University

Motivation

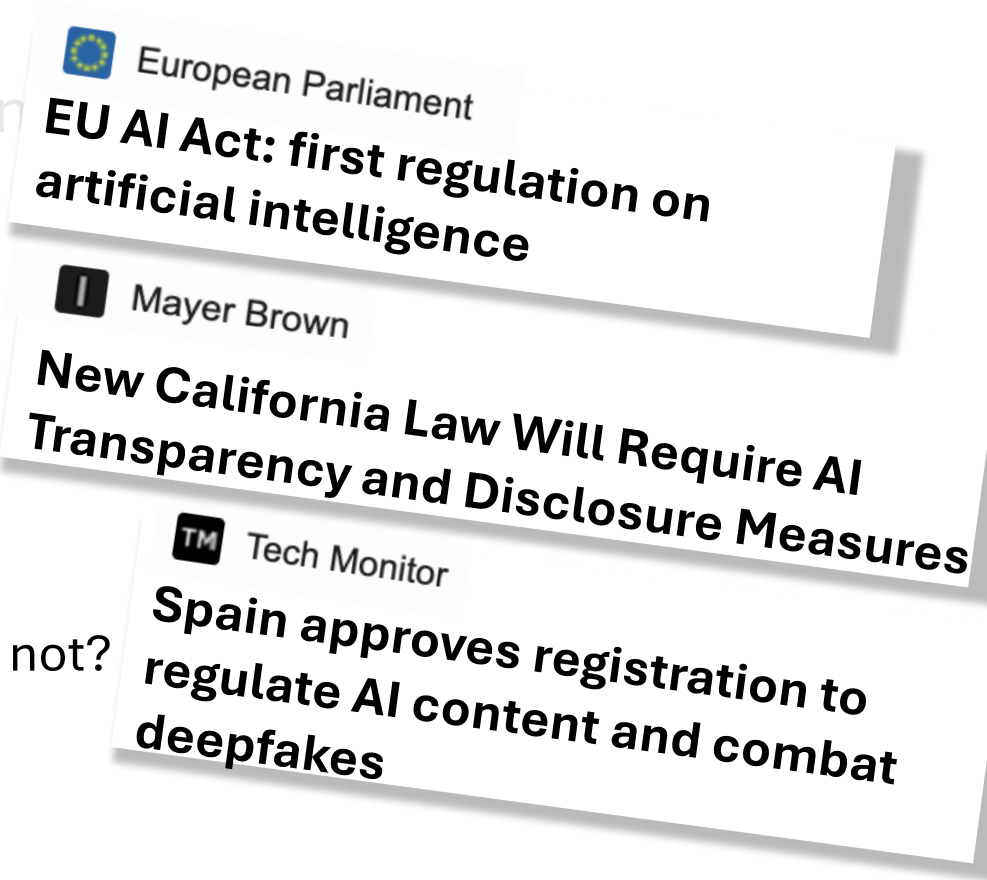
- LLMs demonstrate human-level text generation capabilities
- **Prone to misuse**
  - Misinformation & Fake news
  - AI-generated scams
  - Academic dishonesty



Motivation

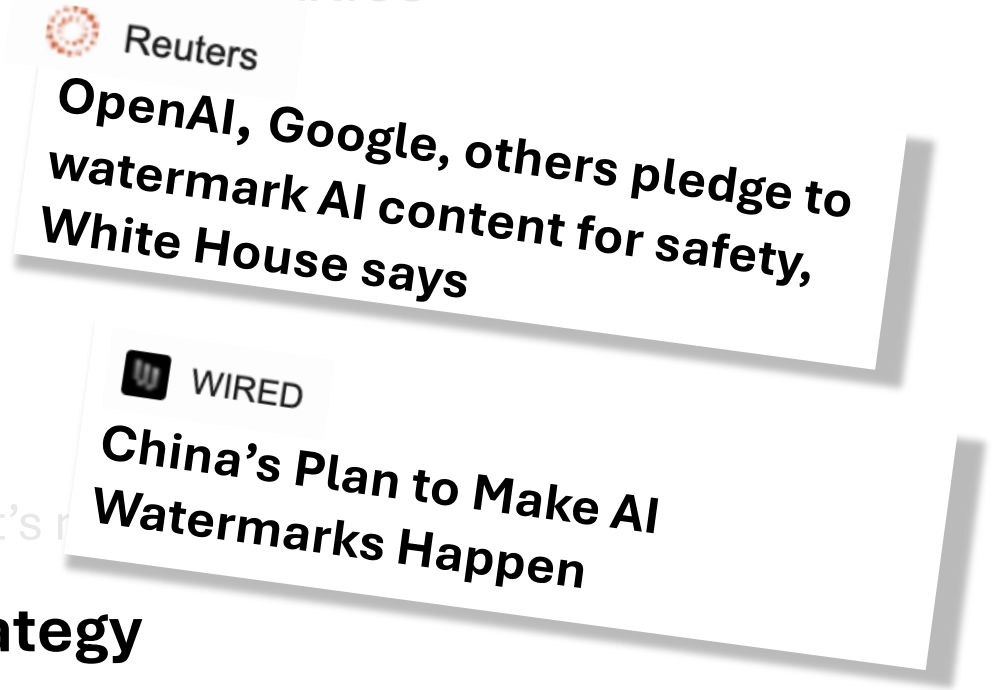
- LLMs demonstrate human-level text generation
- **Prone to misuse**
  - Misinformation & Fake news
  - AI-generated scams
  - Academic dishonesty

- **Regulations hinge on this question**
  - Can we reliably know what's AI generated and what's not?



Motivation

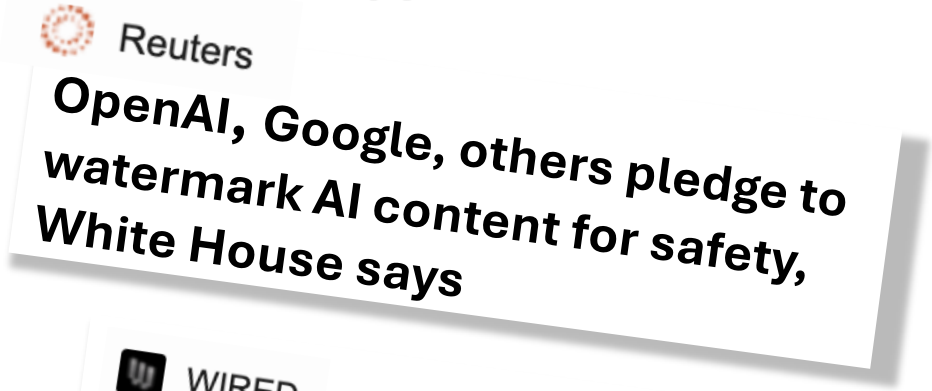
- LLMs demonstrate human-level text generation capabilities
- **Prone to misuse**
  - Misinformation & Fake news
  - AI-generated scams
  - Academic dishonesty
- **Regulations hinge on this question**
  - Can we reliably know what's AI generated and what's n



- **Watermarking emerges as a promising strategy**

Motivation

- LLMs demonstrate human-level text generation capabilities
- **Prone to misuse**
  - Misinformation & Fake news
  - AI-generated scams
  - Academic dishonesty
- **Regulations hinge on this question**
  - Can we reliably know what's AI generated and what's r



- **Watermarking emerged as a promising strategy**

How can we reliably embed & detect watermarks?

Watermarking LLMs

- Embedding a watermark signal into the token **distribution**

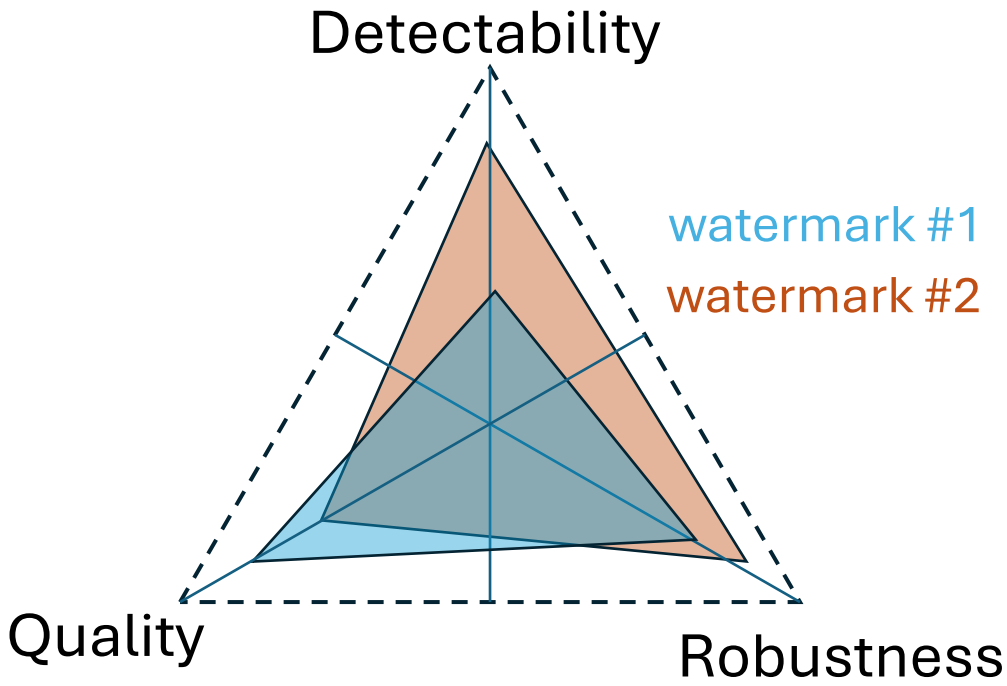
Watermarking LLMs

- Embedding a watermark signal into the token distribution

- Myriad watermarking schemes
  - Red-Green (Kirchenbauer et. al. 23)
  - Inverse transform (Kuditipudi et. al. 23)
  - Gumbel-max (Aaronson & Kircher 23)
  - Error correction codes (Christ & Gunn 24)
  - Synth-ID (Dathathri 24)
  - He et. al 24
  - …

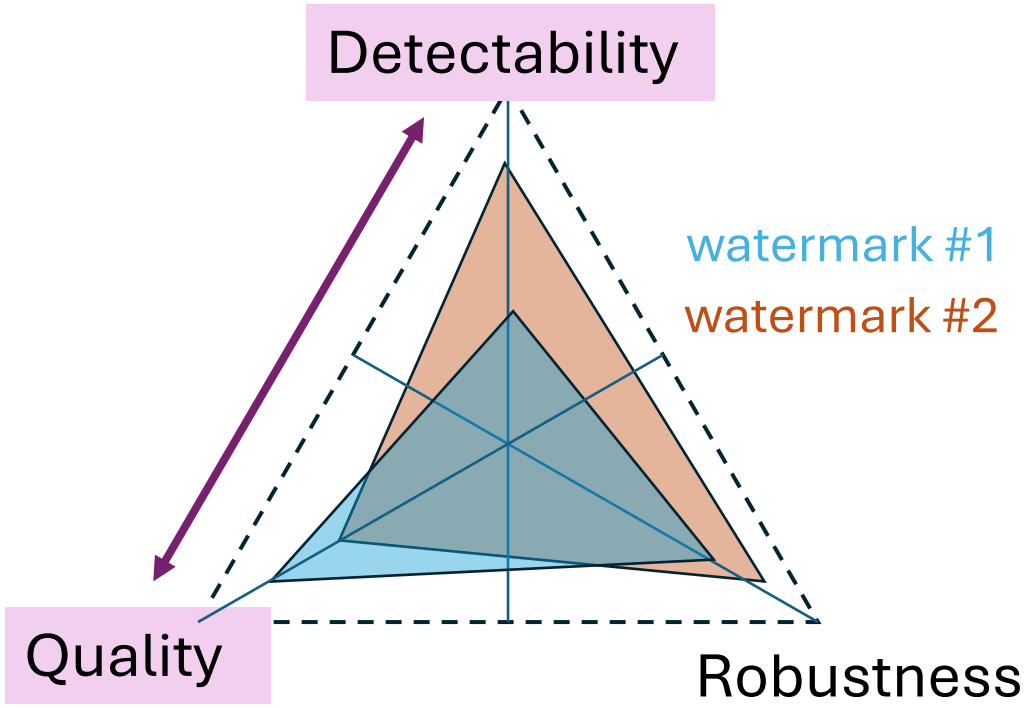
Watermarking LLMs

- Embedding a watermark signal into the token distribution
- Myriad watermarking schemes
  - Red-Green (Kirchenbauer et. al. 23)
  - Inverse transform (Kuditipudi et. al. 23)
  - Gumbel-max (Aaronson & Kircher 23)
  - Error correction codes (Christ & Gunn 24)
  - Synth-ID (Dathathri 24)
  - He et. al 24
  - …
- Trade-off between desirable properties



Watermarking LLMs

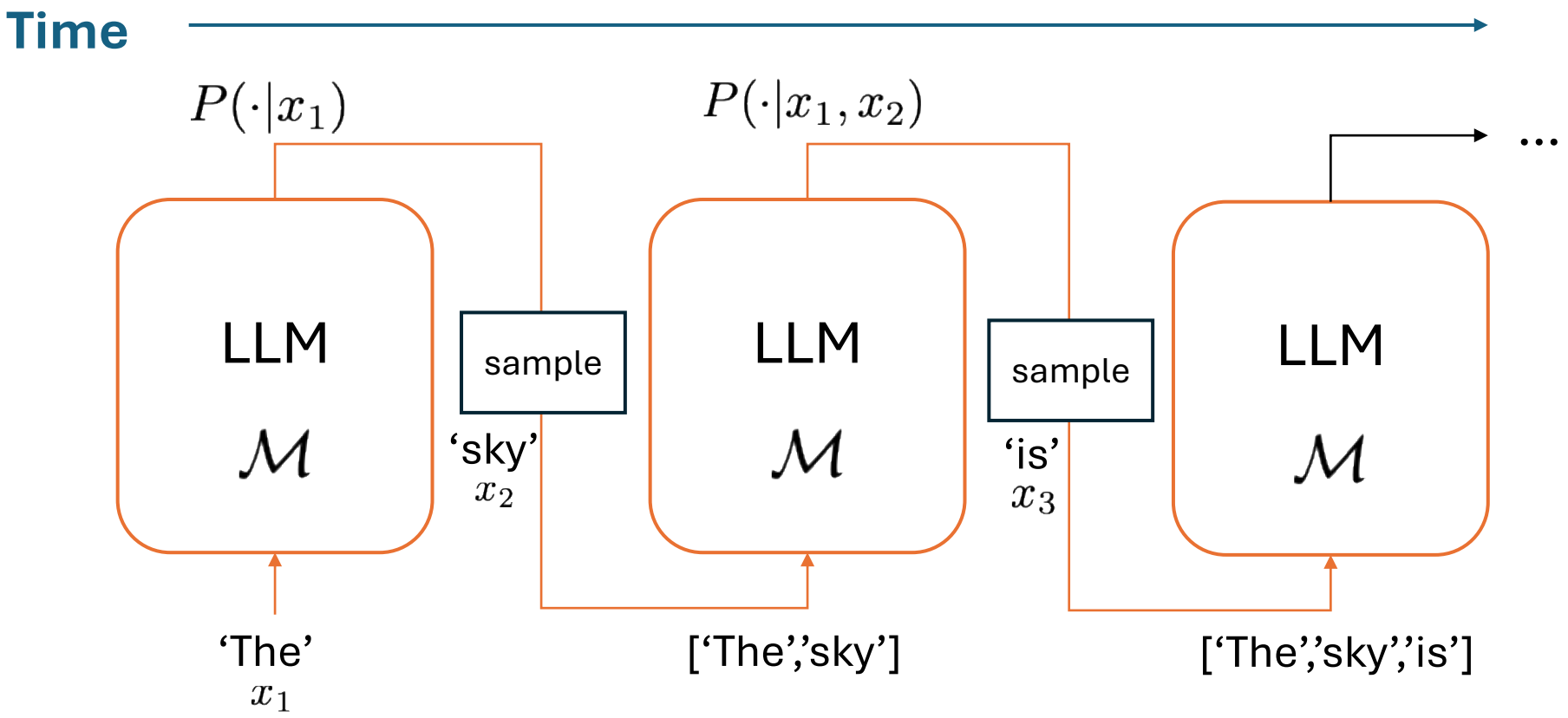
- Embedding a watermark signal into the token distribution
- Myriad watermarking schemes
  - Red-Green (Kirchenbauer et. al. 23)
  - Inverse transform (Kuditipudi et. al. 23)
  - Gumbel-max (Aaronson & Kircher 23)
  - Error correction codes (Christ & Gunn 24)
  - Synth-ID (Dathathri 24)
  - He et. al 24
  - ...
- Trade-off between desirable properties



- We optimize for the **quality-detectability** trade off.

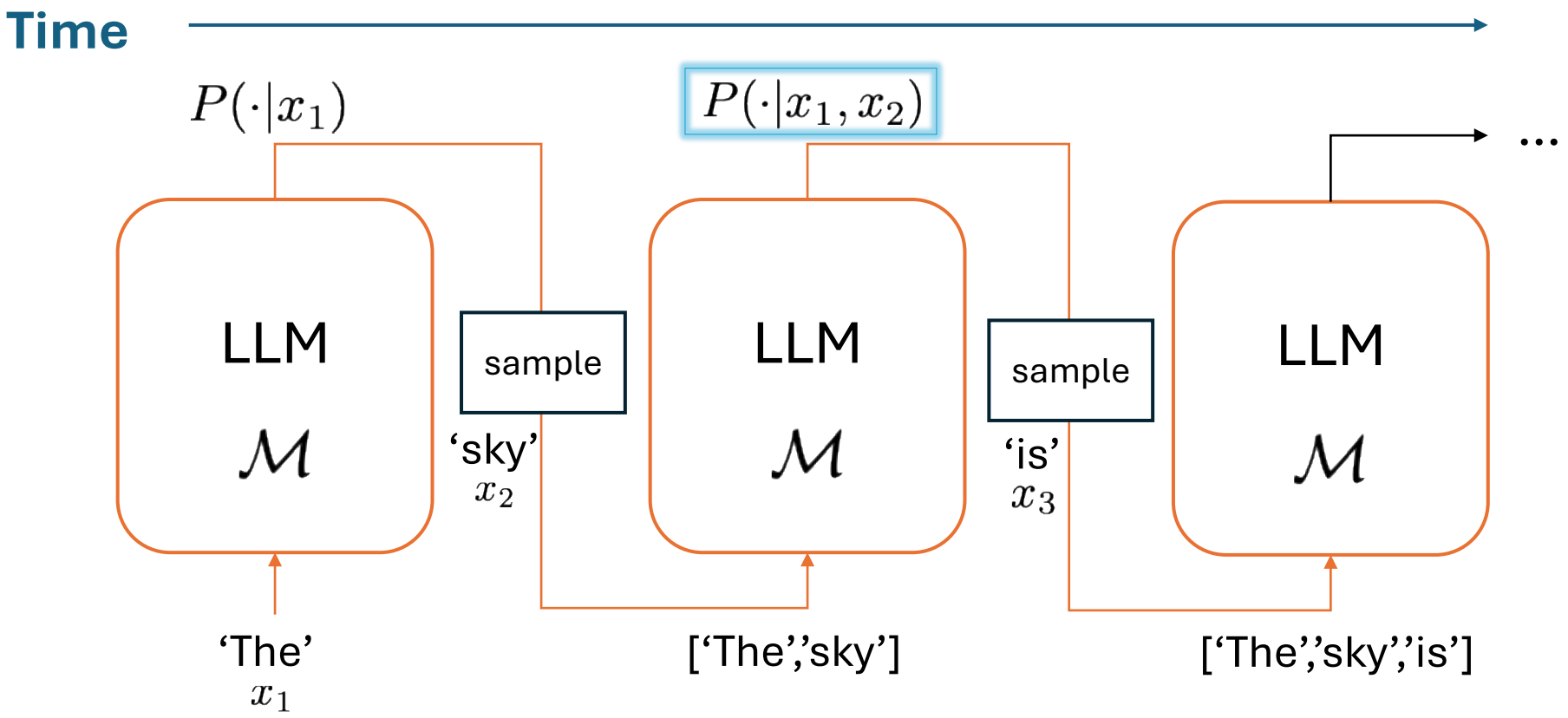
LLM Text Generation

- Tokens, vocabulary $\mathcal{X}$, $|\mathcal{X}| = m$
- Autoregressive token generation



LLM Text Generation

- We focus on **token-level watermarking**
  - Treat each distribution independently



What is LLM watermarking about?

- White-box watermarks
  - **Access** to LLM distribution $P(\cdot|x_1, \ldots, x_{i-1})$
  - **No access** to model weights

What is LLM watermarking about?

- White-box watermarks
  - **Access** to LLM distribution $P(\cdot|x_1, \ldots, x_{i-1})$
  - **No access** to model weights

Goal: Embed a watermark into the token distribution

What is LLM watermarking about?

- White-box watermarks
  - **Access** to LLM distribution
  - **No access** to model weights

Goal: Embed a watermark into the token distribution

- **Requirements**:
  - Watermark can be detected from the generated tokens.
  - Textual quality is not affected.

What is LLM watermarking about?

- White-box watermarks
  - **Access** to LLM distribution
  - **No access** to model weights

Goal: Embed a watermark into the token distribution

- **Requirements**:
  - Watermark can be detected from the generated tokens.
  - Textual quality is not affected.

Detection – How easy it is to detect the watermark from the text

What is LLM watermarking about?

- White-box watermarks
  - **Access** to LLM distribution
  - **No access** to model weights

Goal: Embed a watermark into the token distribution

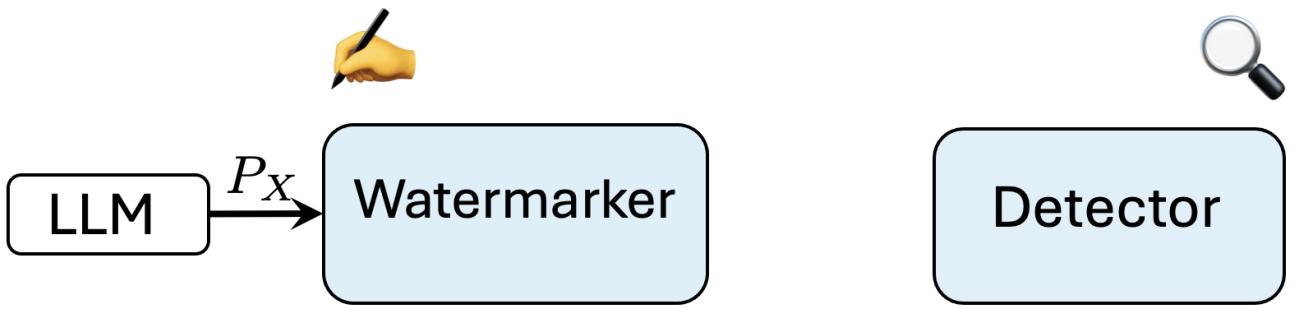
- **Requirements**:
  - Watermark can be detected from the generated tokens.
  - Textual quality is not affected.

Detection – How easy it is to detect the watermark from the text

Quality – Distance from original token distribution, distortion

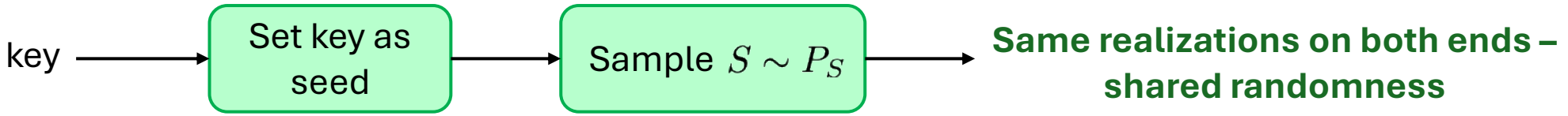
Towards a Formulation

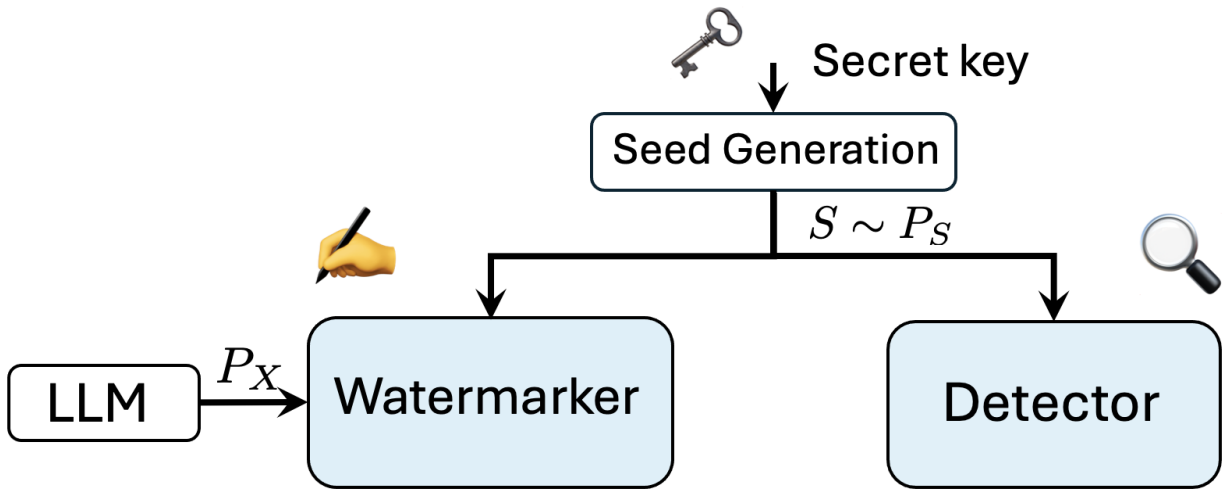
- Two parties – *Watermarker* and *Detector*



First Step Towards a Formulation

- Both parties share **secret key** – generate **Shared randomness**
  - Shared side information $S \sim P_S$, we consider uniform over $\mathcal{S} = [0 : k-1]$



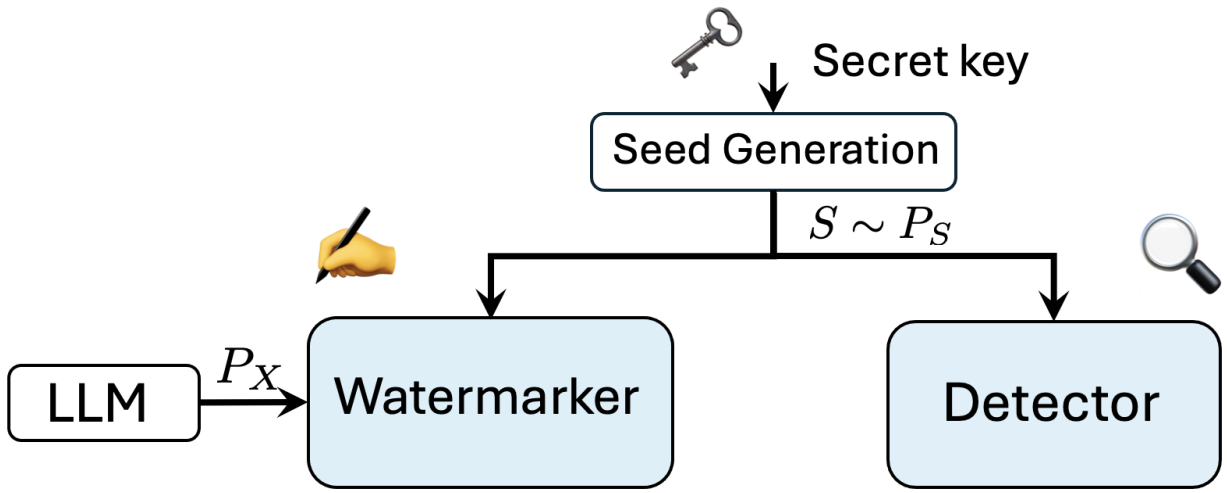


Towards a Formulation

- **Score function** $f(x, s)$

token $x \in \mathcal{X}$

Shared randomness $s \in \mathcal{S}$

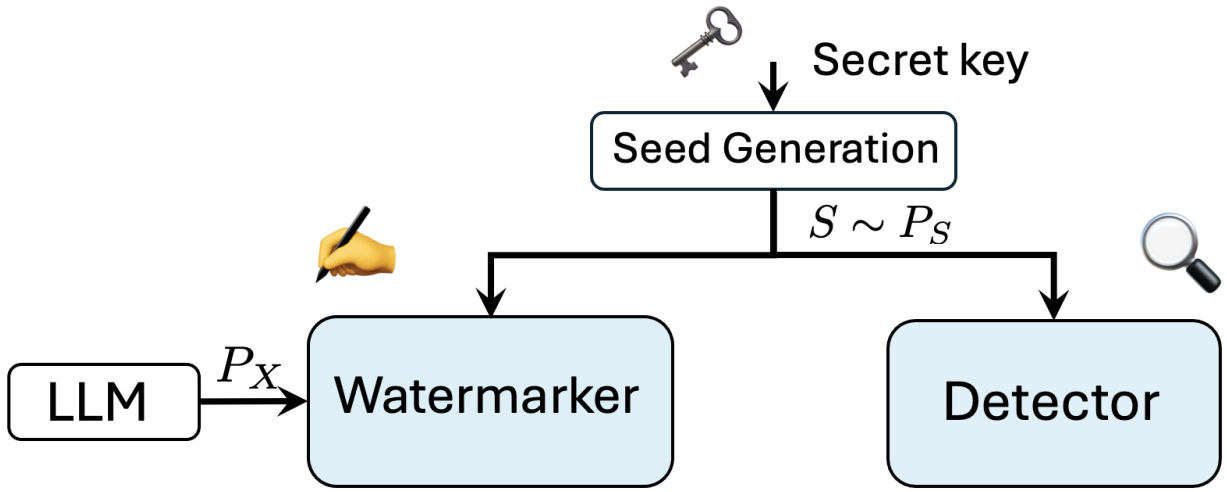


Towards a Formulation

token $x \in \mathcal{X}$

Shared randomness $s \in \mathcal{S}$

- Score function $f(x, s)$
  - **Watermarker** – Generate *watermarked distribution* $P_{X|S}$

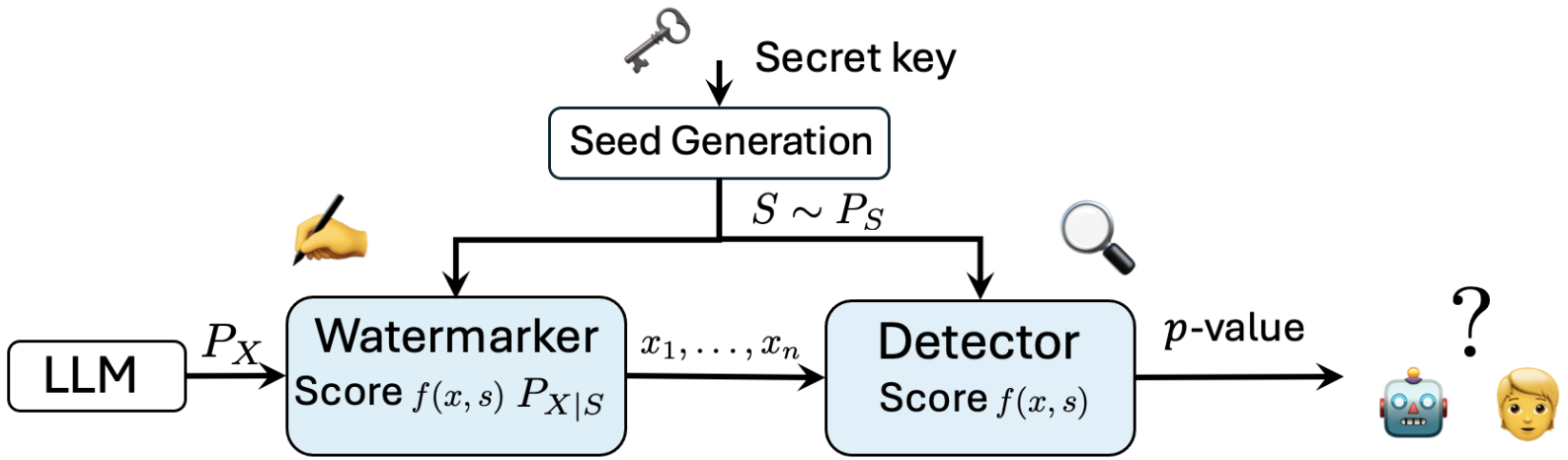


Towards a Formulation

token $x \in \mathcal{X}$

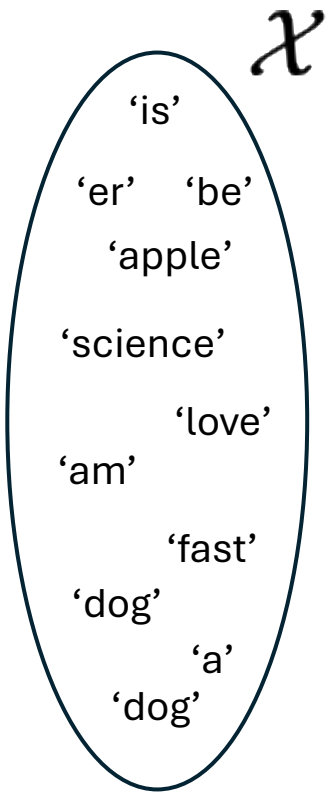
Shared randomness $s \in \mathcal{S}$

- Score function $f(x, s)$
  - Watermarker – Generate *watermarked distribution* $P_{X|S}$
  - **Detector** – Apply a statistical test $\frac{1}{n}\sum_{t=1}^{n} f(x_t, s_t) \geq \tau$



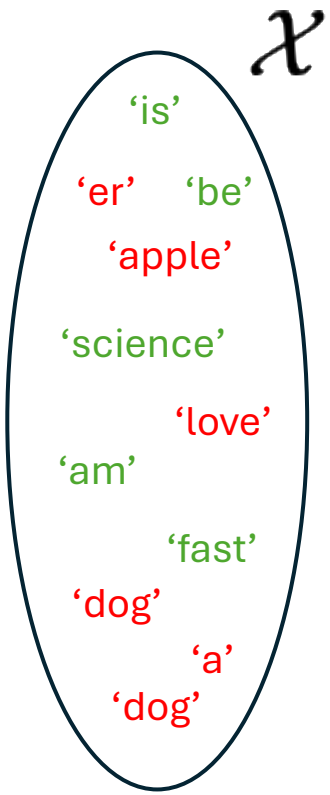
Example – Red Green Watermark (Kirchenbauer et. al. 23')

- **Key idea:** Both parties share a partition of token vocabulary $\mathcal{X}$



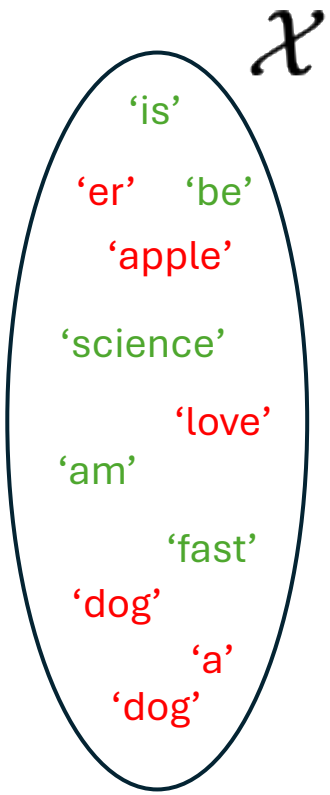
Example – Red Green Watermark (Kirchenbauer et. al. 23')

- **Key idea:** Both parties share a partition of token vocabulary $\mathcal{X}$
- Binary partition – Red list and Green list
  - Random partition – according to shared randomness s



Example – Red Green Watermark (Kirchenbauer et. al. 23')

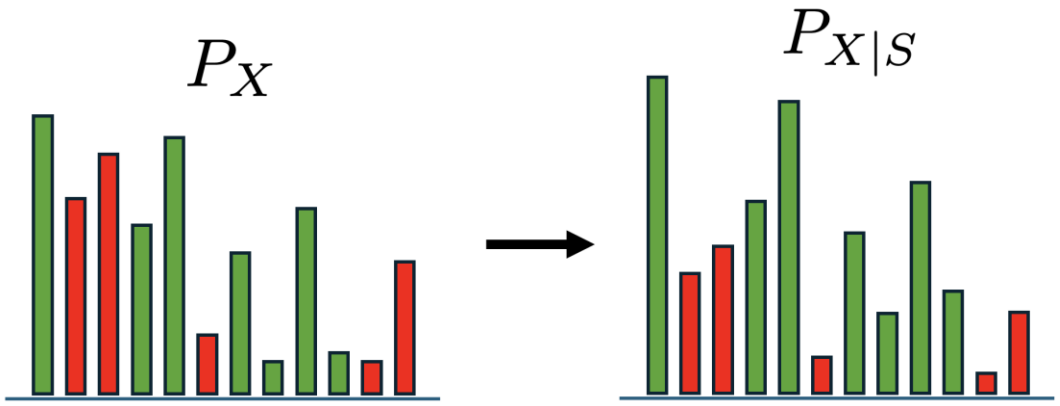
- **Key idea:** Both parties share a partition of $\mathcal{X}$
- Binary partition – Red list and Green list
  - Random partition – according to shared randomness s
  - **Score function** – green list membership

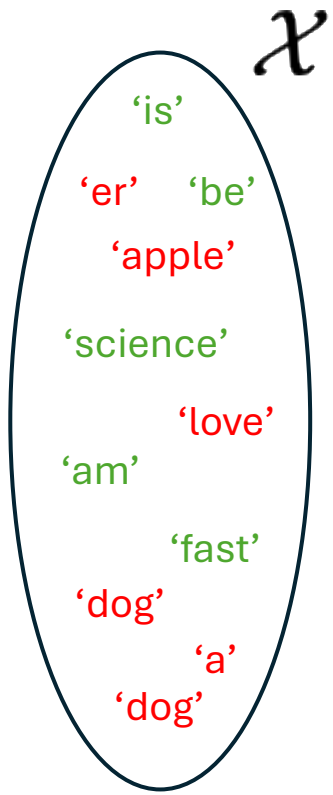


$$f(x, s) = \mathbf{1}\{x \in \text{GreenList}(s)\}$$

Example – Red Green Watermark (Kirchenbauer et. al. 23')

- **Key idea:** Both parties share a partition of $\mathcal{X}$
- Binary partition – Red list and Green list
  - Random partition – according to shared randomness s
  - Score function – green list membership
- Watermarking - Reweigh token distribution
  - **Increase** the probability of green list tokens
  - **Decrease** the probability of red list tokens.





$$f(x,s) = \mathbf{1}\{x \in \text{GreenList}(s)\}$$

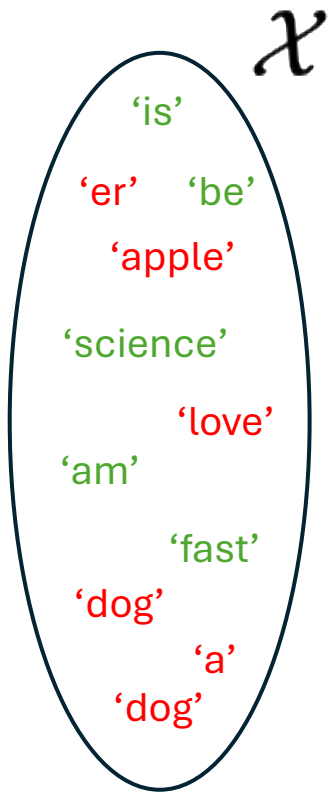
$$P_{X|S=s} = \frac{P_X(x)e^{1+\delta f(x,s)}}{\sum_{x' \in \mathbb{X}} P_X(x')e^{1+\delta f(x,s)}}$$

Example – Red Green Watermark (Kirchenbauer et. al. 23')

- Statistical test – count how many green tokens
  - Detector generates same s samples.

$$\frac{1}{n}\sum_{t=1}^{n}\mathbf{1}\{x_t \in \mathsf{GreenList}(s_t)\} \geq \tau$$

- Test LHS is a proxy for $\mathbb{E}\left[f(X,S)\right]$.



$$f(x,s) = \mathbf{1}\{x \in \mathsf{GreenList}(s)\}$$

$$P_{X|S=s} = \frac{P_X(x)e^{1+\delta f(x,s)}}{\sum_{x' \in \mathbb{X}} P_X(x')e^{1+\delta f(x,s)}}$$

Low Entropy Text Generation

- Generated tokens are relatively deterministic
  - Coding $P(\text{'world'} \mid \text{'print("hello '})$
  - Math $P(\text{'3'} \mid \text{'1 + 2 ='})$

Low Entropy Text Generation

- Generated tokens are relatively deterministic
  - Coding $P(\text{'world'} \mid \text{'print("hello '})$
  - Math $P(\text{'3'} \mid \text{'1 + 2 ='})$

- **Model**: min-entropy constraint $P_\lambda = \left\{ P \in \Delta_m : \max_x P(x) \leq \lambda \right\}$

$\lambda = \frac{1}{m}$ → Uniform

$\lambda = 1$ → includes one hot

Low Entropy Text Generation

- Generated tokens are relatively deterministic
  - Coding $P(\text{'world'} \mid \text{'print("hello '})$
  - Math $P(\text{'3'} \mid \text{'1 + 2 ='})$

- **Model**: min-entropy constraint $P_\lambda = \left\{ P \in \Delta_m : \max_x P(x) \leq \lambda \right\}$

$\lambda = \frac{1}{m}$ → Uniform

$\lambda = 1$ → includes one hot

- This work: $\lambda \in \left[\frac{1}{2}, 1\right]$
  - Worst case: A single token gets a probability of 0.5
  - Spiky distributions – Token vocabularies are very big.

Low Entropy Text Generation

- Generated tokens are relatively deterministic
  - Coding $P(\text{'world'} \mid \text{'print("hello '})$
  - Math $P(\text{'3'} \mid \text{'1 + 2 ='})$

- **Model**: min-entropy constraint $P_\lambda = \left\{ P \in \Delta_m : \max_x P(x) \leq \lambda \right\}$

$\lambda = \frac{1}{m}$ → Uniform

$\lambda = 1$ → includes one hot

- This work: $\lambda \in \left[\frac{1}{2}, 1\right]$
  - Worst case: A single token gets a probability of 0.5
  - Spiky distributions – Token vocabularies are very big.

*** Are all texts potentially low-entropy?**

Optimization Problem Formulation - Objective

- Recall:
  - Shared randomness s
  - Score function $f(x, s)$
  - Watermarked distribution $P_{X|S}$

Optimization Problem Formulation - Objective

- Recall:
  - Shared randomness s
  - Score function $f(x, s)$
  - Watermarked distribution $P_{X|S}$

When X is *watermarked*: $(X, S) \sim P_S P_{X|S}$

Watermarked distribution

When X is *not watermarked*: $(X, S) \sim P_S P_X$

Original distribution

Optimization Problem Formulation - Objective

When X is *watermarked*: $(X, S) \sim P_S P_{X|S}$ → Watermarked distribution

When X is *not watermarked*: $(X, S) \sim P_S P_X$ → Original distribution

- Recall - Proxy for detection - Expected score: $\mathbb{E}\left[f(X, S)\right]$
- Objective function - maximize the gap between expected scores:

$$\mathbb{E}_{P_S P_{X|S}}[f(X, S)] - \mathbb{E}_{P_S P_X}[f(X, S)]$$

Our contribution: Optimization Problem Formulation

- Considerations:

$$\mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Our contribution: Optimization Problem Formulation

- Considerations:
  - **Watermarked distribution** chosen after $f(x,s)$ and P_X are set

$$\max_{P_{X,S}} \quad \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

$$(P_{X|S})$$

Our contribution: Optimization Problem Formulation

- Considerations:
  - Watermarked distribution chosen after $f(x, s)$ and P_X are set
  - No control over **token distribution**

$$\min_{P_X \in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X, S)] - \mathbb{E}_{P_S P_X}[f(X, S)]$$

Our contribution: Optimization Problem Formulation

- Considerations:
  - Watermarked distribution chosen after score and token are set
  - No control over token distribution
  - **Score** set prior to 'communication' – can't be dependent on anything

$$\max_{f \in \mathcal{F}} \min_{P_X \in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Optimization Formulation – Zero Distortion

- Zero distortion watermarks: $\mathbb{E}_{P_S}[P_{X|S}] = P_X$
  - Proxy for textual quality
  - Proxy for eavesdropper undetectability

$$\max_{f \in \mathcal{F}} \min_{P_X \in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Optimization Formulation – Zero Distortion

- Zero distortion watermarks: $\mathbb{E}_{P_S}[P_{X|S}] = P_X$
  - Proxy for textual quality
  - Proxy for eavesdropper undetectability

- Get that ‘for free’ by optimizing over **couplings** of (P_X, P_S)

$$\left\{ P_{X,S} \middle| \sum_{x \in \mathcal{X}} P_{X,S} = P_S, \sum_{s \in \mathcal{S}} P_{X,S} = P_X \right\}$$

$$\max_{f \in \mathcal{F}} \min_{P_X \in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Optimization Formulation – Zero Distortion

- Zero distortion watermarks: $\mathbb{E}_{P_S}[P_{X|S}] = P_X$
  - Proxy for textual quality
  - Proxy for eavesdropper undetectability

- Get that ‘for free’ by optimizing over **couplings** of (P_X, P_S)

$$\left\{ P_{X,S} \middle| \sum_{x \in \mathcal{X}} P_{X,S} = P_S, \sum_{s \in \mathcal{S}} P_{X,S} = P_X \right\}$$

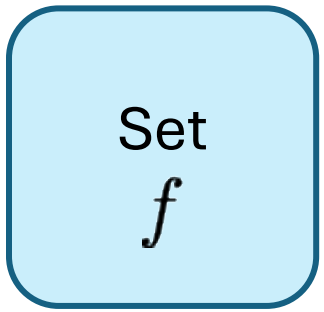
Optimal Transport?

$$\max_{f \in \mathcal{F}} \min_{P_X \in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Proposed Method

- Set score function $f \in \mathcal{F}$

Prior to watermarking

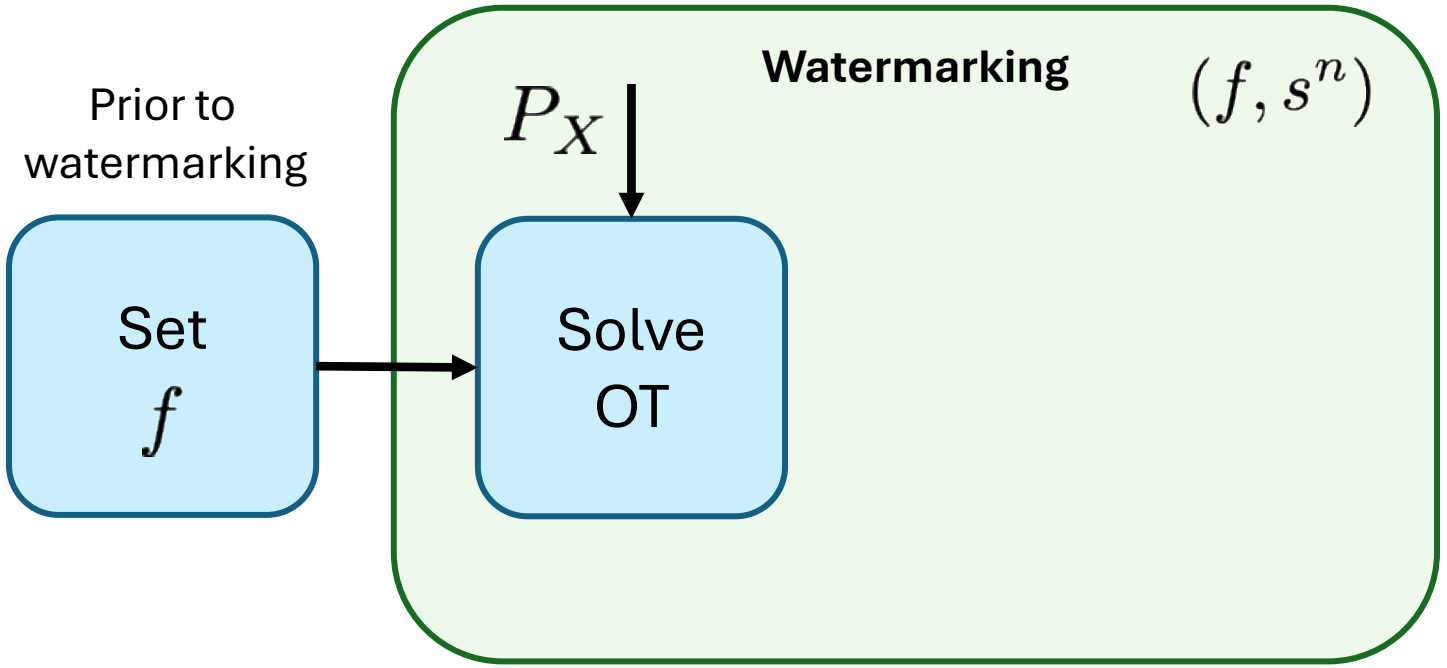


Proposed Method

- Given (f, P_X) solve the Optimal Transport problem:

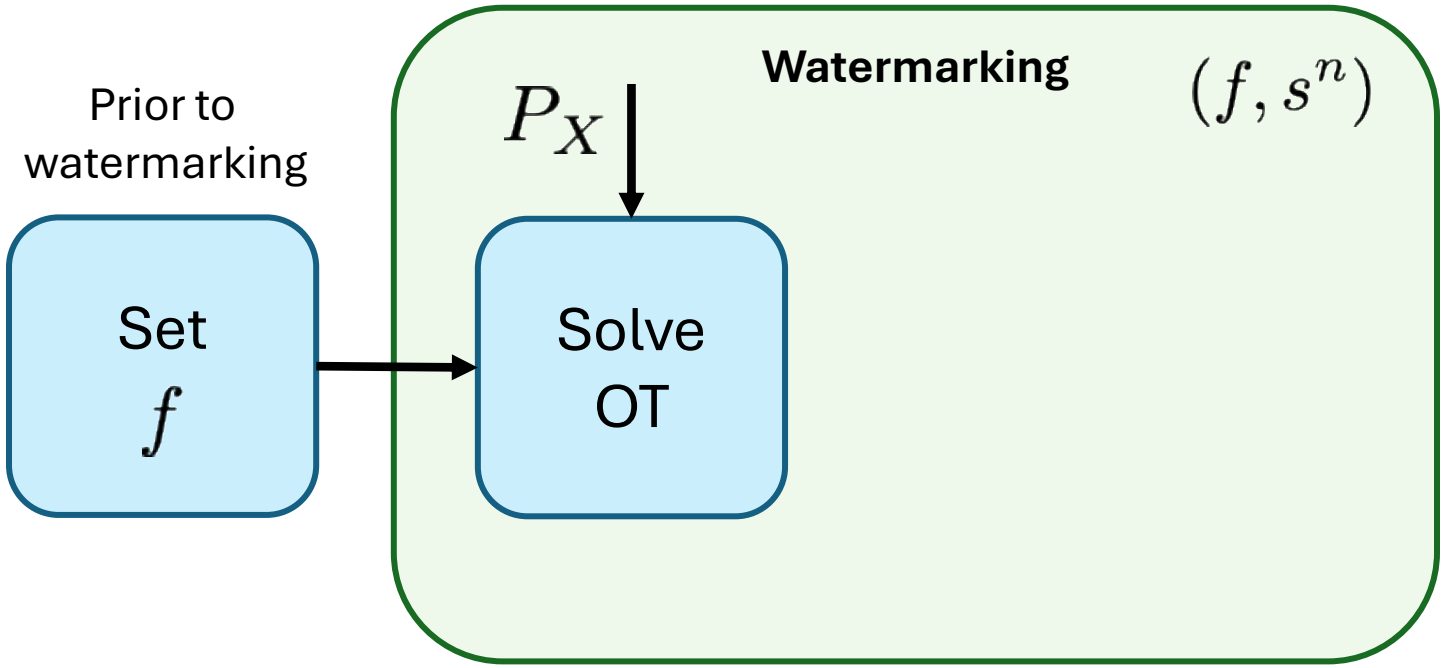
$$\max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Constant w.r.t coupling



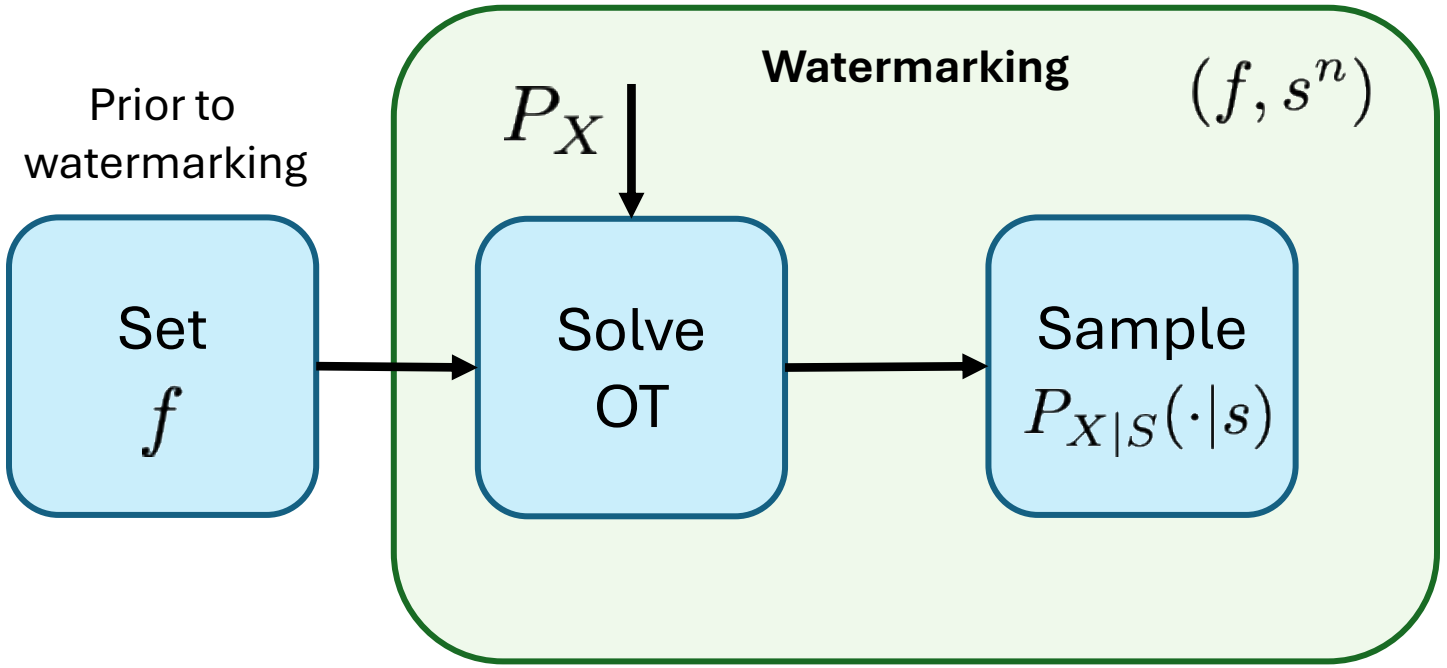
Proposed Method

- Given (f, P_X) solve the Optimal Transport problem:
- Sinkhorn $\max_{P_{X,S}} \mathbb{E}[f(X,S)]$ f Is the OT cost



Proposed Method

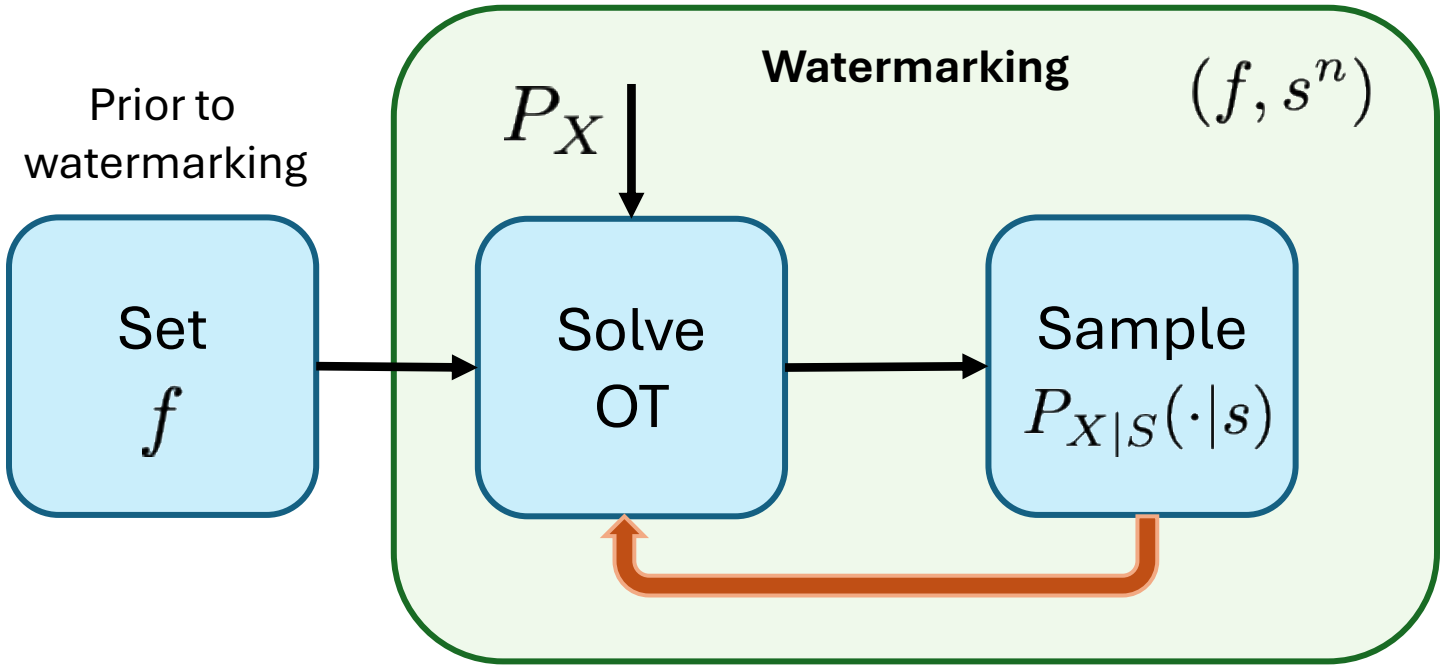
- Extract $P_{X|S}(\cdot|s)$ from Optimal coupling – normalized column.
- Sample $P_{X|S}(\cdot|s)$ to obtain a watermarked token



Proposed Method

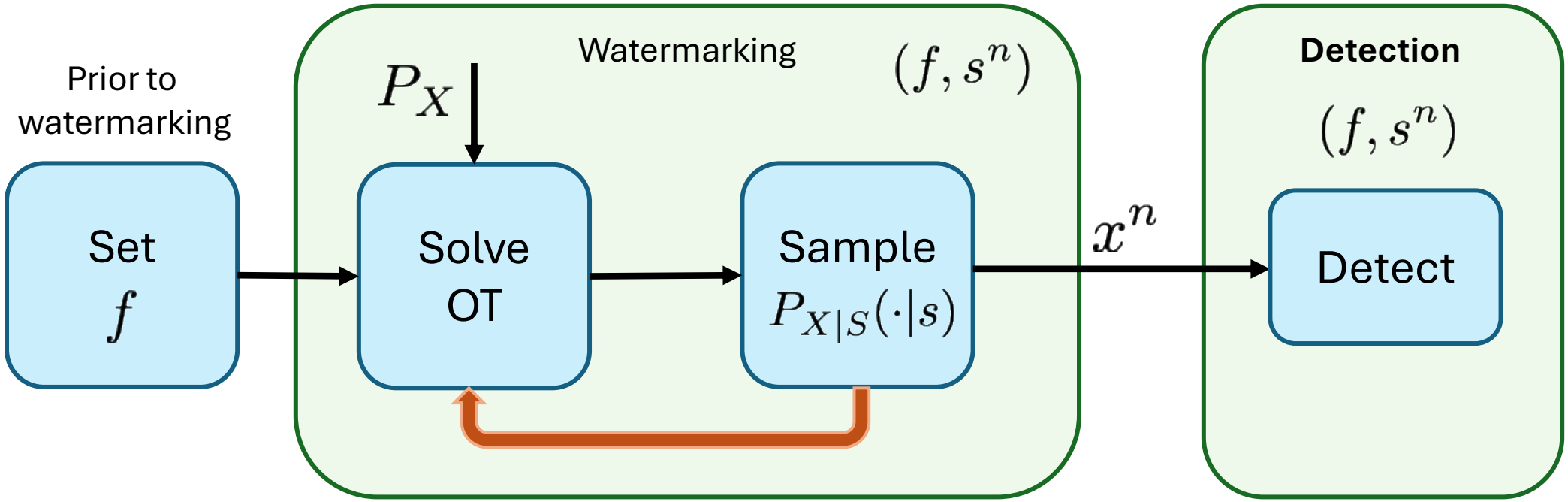
- Extract $P_{X|S}(\cdot|s)$ from Optimal coupling – column.
- Sample $P_{X|S}(\cdot|s)$ to obtained watermarked token

Repeat for n tokens



Proposed Method

- **Detector:**
  - Knows f and performs statistical test: $\frac{1}{n}\sum_{t=1}^{n} f(x_t, s_t) \geq \tau$



Summary Until Now

- Optimization Problem Formulation:

$$\max_{f \in \mathcal{F}} \quad \min_{P_X \in P_\lambda} \quad \max_{P_{X,S}} \quad \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Summary Until Now

- Optimization Problem Formulation:

$$\max_{f \in \mathcal{F}} \min_{P_X \in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

- Solving an OT via Sinkhorn
  - Score f determines the OT cost
  - Provides a zero-distortion watermark

Summary Until Now

- Optimization Problem Formulation:

$$\mathsf{D}_{\mathsf{gap}}(m,k,\lambda,\mathcal{F}) = \max_{f\in\mathcal{F}} \min_{P_X\in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

- Solving an OT via Sinkhorn
  - Score f determines the OT cost
  - Provides a zero-distortion watermark

- We call the solution the ***detection gap*** $\mathsf{D}_{\mathsf{gap}}(m,k,\lambda,\mathcal{F})$

Summary Until Now

- Optimization Problem Formulation:

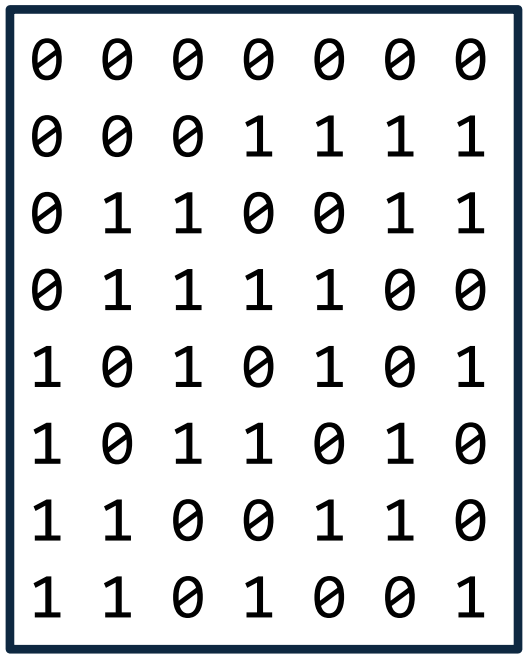
$$\mathsf{D}_{\mathsf{gap}}(m,k,\lambda,\mathcal{F}) = \max_{f\in\mathcal{F}} \min_{P_X\in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_{\cdot}}[f(\cdot$$

- Solving an OT via Sinkhorn
  - Score f determines the O
  - Provides a z

How should we design the score function?

- …ult the solution the ***detection gap*** $\mathsf{D}_{\mathsf{gap}}(m,k,\lambda,\mathcal{F})$

`SimplexWater`
Binary Score Functions



Binary Scores

Token vocabulary $\mathcal{X}$ Shared randomness $\mathcal{S}$

- Score class $\mathcal{F}_{\text{bin}} = \{f : [1:m] \times [1:k] \to \{0,1\}\}$
  - Example – red/green

Binary Scores

$$\max_{f\in\mathcal{F}} \min_{P_X\in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Token vocabulary Shared randomness

- Score class $\mathcal{F}_{\text{bin}} = \{f : [1:m] \times [1:k] \to \{0,1\}\}$
  - Example – red/green
- Optimization can be **simplified**:

Proposition 1. *Let $\lambda \in \left[\frac{1}{2}, 1\right)$. For $f \in \mathcal{F}_{\text{bin}}$, define the vector $f_i = [f(i,1), \ldots, f(i,k)] \in \{0,1\}^k$ for each $i \in \mathcal{X}$. Then,*

$$\mathsf{D}_{\text{gap}}(m,k,\lambda,\mathcal{F}_{\text{bin}}) = \max_{f\in\mathcal{F}_{\text{bin}}} \min_{i,j\in\mathcal{X}, i\neq j} \frac{(1-\lambda) d_H(f_i,f_j)}{k},$$

where $d_H(a,b) = \sum_{i=1}^{k} \mathbf{1}_{\{a_i \neq b_i\}}$ denotes the Hamming distance between $a, b \in \{0,1\}^k$ and $\mathbf{1}_{\{\cdot\}}$ is the indicator function.

Binary Scores

$$\max_{f\in\mathcal{F}} \min_{P_X\in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Token vocabulary Shared randomness

- Score class $\mathcal{F}_{\text{bin}} = \{f : [1:m] \times [1:k] \to \{0,1\}\}$
  - Example – red/green
- Optimization can be **simplified**:

Proposition 1. *Let $\lambda \in \left[\frac{1}{2}, 1\right)$. For $f \in \mathcal{F}_{\text{bin}}$, define the vector $f_i = [f(i,1), \ldots, f(i,k)] \in \{0,1\}^k$ for each $i \in \mathcal{X}$. Then,*

$$\mathsf{D}_{\text{gap}}(m,k,\lambda,\mathcal{F}_{\text{bin}}) = \max_{f\in\mathcal{F}_{\text{bin}}} \min_{i,j\in\mathcal{X}, i\neq j} \frac{(1-\lambda) d_H(f_i,f_j)}{k},$$

where $d_H(a,b) = \sum_{i=1}^{k} \mathbf{1}_{\{a_i \neq b_i\}}$ denotes the Hamming distance between $a, b \in \{0,1\}^k$ and $\mathbf{1}_{\{\cdot\}}$ is the indicator function.

Each element in the vocabulary is assigned with a binary vector

Binary Scores

$$\max_{f\in\mathcal{F}} \min_{P_X\in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

- Score class $\mathcal{F}_{\text{bin}} = \{f : [1:m] \times [1:k] \to \{0,1\}\}$
  - Example – red/green
- Optimization can be simplified:

Maximize the *minimum* Hamming distance

Proposition 1. *Let $\lambda \in [\frac{1}{2}, 1)$. For $f \in \mathcal{F}_{\text{bin}}$, define the vector $f_i = [f(i,1), \ldots, f(i,k)] \in \{0,1\}^k$ for each $i \in \mathcal{X}$. Then,*

$$\mathsf{D}_{\text{gap}}(m,k,\lambda,\mathcal{F}_{\text{bin}}) = \max_{f\in\mathcal{F}_{\text{bin}}} \min_{i,j\in\mathcal{X}, i\neq j} \frac{(1-\lambda) d_H(f_i,f_j)}{k},$$

where $d_H(a,b) = \sum_{i=1}^k \mathbf{1}_{\{a_i \neq b_i\}}$ denotes the Hamming distance between $a, b \in \{0,1\}^k$ and $\mathbf{1}_{\{\cdot\}}$ is the indicator function.

Each element in the vocabulary is assigned with a binary vector

Binary Scores

- This is a coding – theoretic problem!
- f_i are codewords
- We design a distance-maximizing code

Proposition 1. *Let* $\lambda \in \left[\frac{1}{2}, 1\right)$. *For* $f \in \mathcal{F}_{\text{bin}}$, *define the vector* $f_i = [f(i,1), \ldots, f(i,k)] \in \{0,1\}^k$ *for each* $i \in \mathcal{X}$. *Then,*

$$\mathsf{D}_{\text{gap}}(m, k, \lambda, \mathcal{F}_{\text{bin}}) = \max_{f \in \mathcal{F}_{\text{bin}}} \min_{i,j \in \mathcal{X}, i \neq j} \frac{(1-\lambda) d_H(f_i, f_j)}{k},$$

where $d_H(a,b) = \sum_{i=1}^{k} \mathbf{1}_{\{a_i \neq b_i\}}$ *denotes the Hamming distance between* $a, b \in \{0,1\}^k$ *and* $\mathbf{1}_{\{\cdot\}}$ *is the indicator function.*

Binary Scores

Maximizing detection gap

(Watermarking problem)

Designing a distance-maximizing code

(Coding theory problem)

Binary Scores

Maximizing detection gap

(Watermarking problem)

Designing a distance-maximizing code

(Coding theory problem)

```
import coding_theory
```

Binary Scores

Maximizing detection gap ⟺ **Designing a distance-maximizing code**

(Watermarking problem)

(Coding theory problem)

```
import coding_theory
```

Theorem 1 (Maximum Detection Gap Upper Bound). *Consider the class of binary score functions* $\mathcal{F}_{\text{bin}}$ *and uniform* P_S*. Then, for any* $\lambda \in \left[\frac{1}{2}, 1\right)$*, the maximum detection gap can be bounded as*

$$\mathsf{D}_{\text{gap}}(m, k, \lambda, \mathcal{F}_{\text{bin}}) \leq \frac{m(1-\lambda)}{2(m-1)}.$$

λ - Distribution family parameter

m - Vocabulary size

Binary Scores

Maximizing detection gap ⟺ **Designing a distance-maximizing code**

```
import coding_theory
```

Theorem 1 (Maximum Detection Gap Upper Bound). *Consider the class of binary score functions* $\mathcal{F}_{\text{bin}}$ *and uniform* P_S. *Then, for any* $\lambda \in \left[\frac{1}{2}, 1\right)$, *the maximum detection gap can be bounded as*

$$\mathsf{D}_{\text{gap}}(m, k, \lambda, \mathcal{F}_{\text{bin}}) \leq \frac{m(1-\lambda)}{2(m-1)}.$$

Proof - Plotkin Bound! (1960)

Achieving the Detection Bound

- Can we **achieve** the upper bound?
  - Equivalently - Design a bound-achieving code

Achieving the Detection Bound

- Can we achieve the upper bound?
  - Equivalently - Design a bound-achieving code

Yes! – Simplex codes

Shared randomness size

$$(|\mathcal{S}| = k = m - 1)$$

Definition 1 (Simplex Code). *For any* $x, s \in [0 : m-1]$, *let* $\mathsf{bin}(x)$, $\mathsf{bin}(s)$ *denote their binary representations respectively using* $\log_2 m$ *bits. A simplex code* $f_{\mathsf{sim}} : [0 : m-1] \times [1 : m-1] \to \{0, 1\}$ *is characterized by*

$$f_{\mathsf{sim}}(x, s) \triangleq \mathrm{dot}(\mathsf{bin}(x), \mathsf{bin}(s)),$$

where $\mathrm{dot}(\mathsf{bin}(x), \mathsf{bin}(s)) \triangleq \sum_{i=1}^{\log_2 m} \mathsf{bin}(x)_i \cdot \mathsf{bin}(s)_i$ *and* $\mathsf{bin}(v)_i$ *denotes the* i*th bit in the binary representation of* v.

Achieving the Detection Bound

- Simplex code

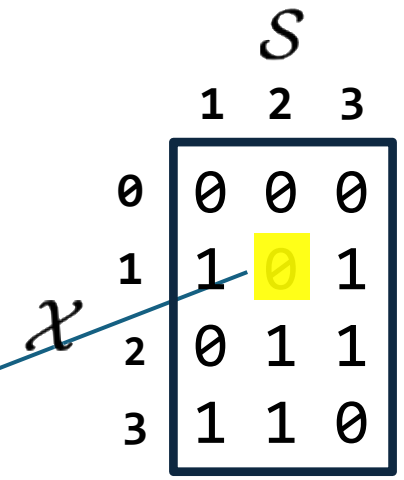
Definition 1 (Simplex Code). *For any* $x, s \in [0 : m-1]$, *let* $\mathsf{bin}(x)$, $\mathsf{bin}(s)$ *denote their binary representations respectively using* $\log_2 m$ *bits. A simplex code* $f_{\mathsf{sim}} : [0 : m-1] \times [1 : m-1] \to \{0, 1\}$ *is characterized by*

$$f_{\mathsf{sim}}(x, s) \triangleq \mathrm{dot}(\mathsf{bin}(x), \mathsf{bin}(s)), \tag{5}$$

where $\mathrm{dot}(\mathsf{bin}(x), \mathsf{bin}(s)) \triangleq \sum_{i=1}^{\log_2 m} \mathsf{bin}(x)_i \cdot \mathsf{bin}(s)_i$ *and* $\mathsf{bin}(v)_i$ *denotes the* i*th bit in the binary representation of* v.

- Example – Simplex code $m = 4, k = m - 1 = 3$

$$f(1,2) = \mathsf{dot}(\mathsf{bin}(1), \mathsf{bin}(2)) = \mathsf{dot}(001, 010) = 0$$



`SimplexWater`

- `SimplexWater` – Take score f to be a simplex code!

SimplexWater

- `SimplexWater` – Take score f to be a simplex code!
- m x (m-1) binary lookup table F

Definition 1 (Simplex Code). *For any $x, s \in [0 : m-1]$, let $\mathsf{bin}(x)$, $\mathsf{bin}(s)$ denote their binary representations respectively using $\log_2 m$ bits. A simplex code $f_{\mathsf{sim}} : [0 : m-1] \times [1 : m-1] \to \{0, 1\}$ is characterized by*

$$f_{\mathsf{sim}}(x, s) \triangleq \mathrm{dot}(\mathsf{bin}(x), \mathsf{bin}(s)), \tag{5}$$

where $\mathrm{dot}(\mathsf{bin}(x), \mathsf{bin}(s)) \triangleq \sum_{i=1}^{\log_2 m} \mathsf{bin}(x)_i \cdot \mathsf{bin}(s)_i$ and $\mathsf{bin}(v)_i$ denotes the ith bit in the binary representation of v.

$\mathcal{S}$

$\mathcal{X}$

	1	2	.	.	.	m-1
1	0	0	0	0	0	0	0	1
2	0	0	0	1	1	1	1	0
3	0	1	1	0	0	1	1	1
	0	1	1	1	1	0	0	0
.	1	0	1	0	1	0	1	1
.	1	0	1	1	0	1	0	0
	1	1	0	0	1	1	0	1
m	1	1	0	1	0	0	1	1

$$F_{i,j} = \mathrm{dot}(\mathsf{bin}(i), \mathsf{bin}(j))$$

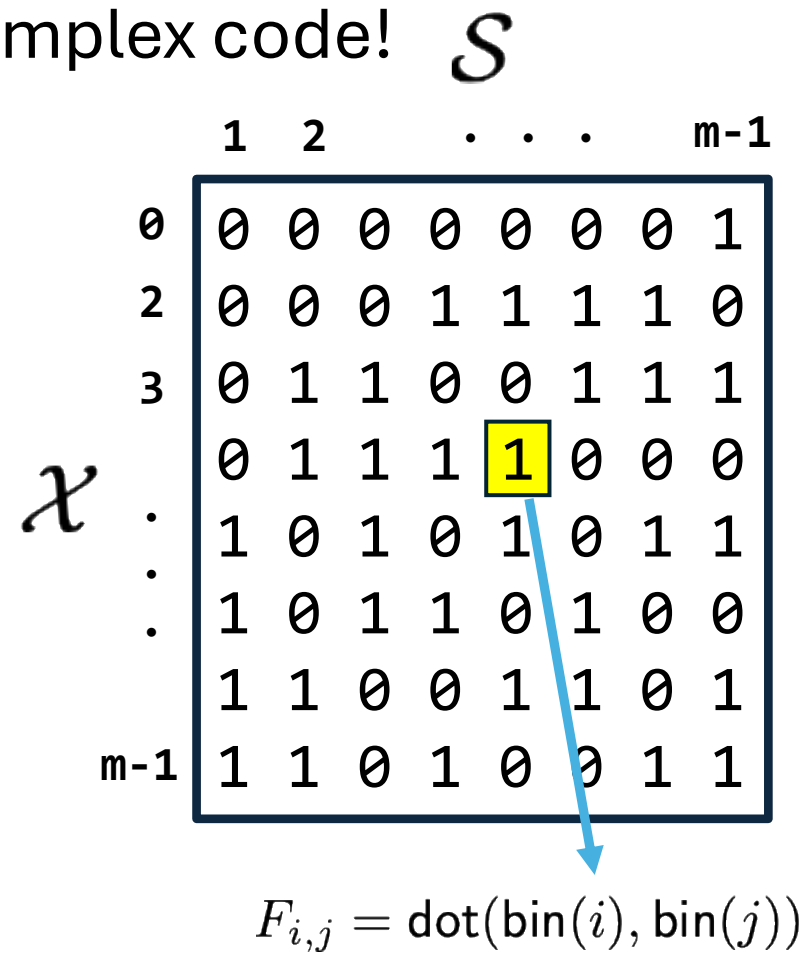
SimplexWater

- `SimplexWater` – Take score f to be a simplex code!
- m x (m-1) binary lookup table F
- Computed apriori on both ends

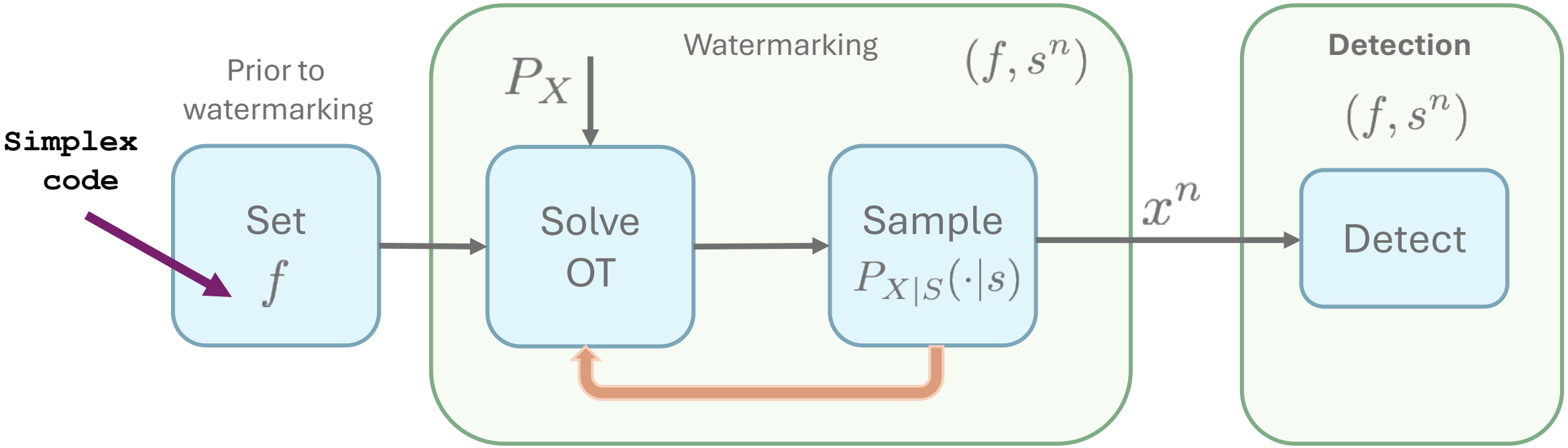
Definition 1 (Simplex Code). *For any $x, s \in [0 : m-1]$, let $\mathsf{bin}(x)$, $\mathsf{bin}(s)$ denote their binary representations respectively using $\log_2 m$ bits. A simplex code $f_{\mathsf{sim}} : [0 : m-1] \times [1 : m-1] \to \{0, 1\}$ is characterized by*

$$f_{\mathsf{sim}}(x, s) \triangleq \mathrm{dot}(\mathsf{bin}(x), \mathsf{bin}(s)), \tag{5}$$

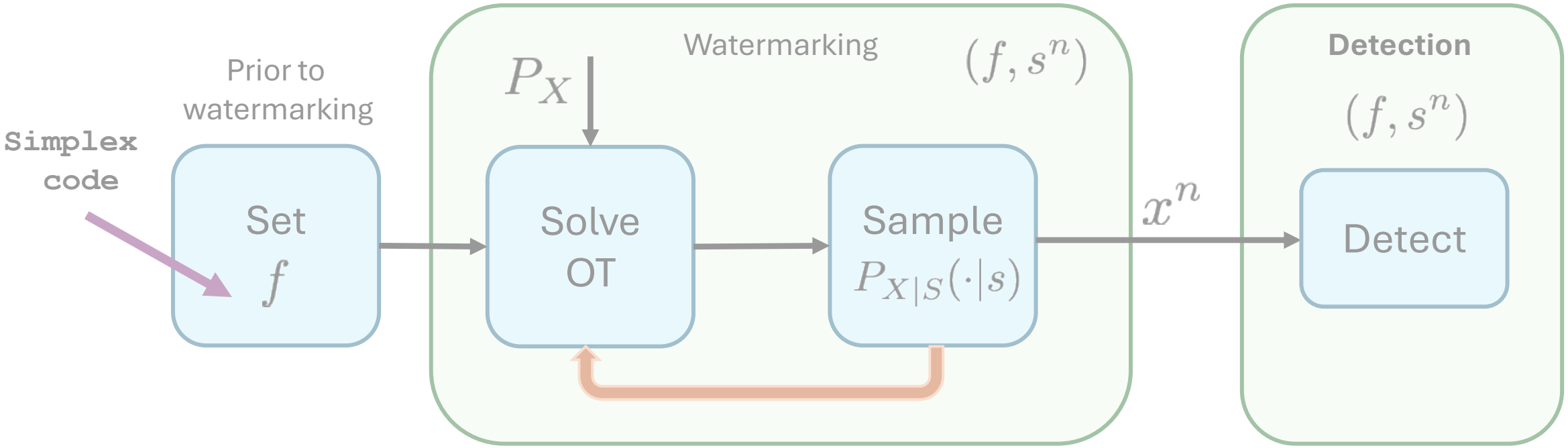
where $\mathrm{dot}(\mathsf{bin}(x), \mathsf{bin}(s)) \triangleq \sum_{i=1}^{\log_2 m} \mathsf{bin}(x)_i \cdot \mathsf{bin}(s)_i$ and $\mathsf{bin}(v)_i$ denotes the ith bit in the binary representation of v.



SimplexWater



SimplexWater



- `SimplexWater` is **optimal** across all binary score watermarks:

Theorem 2 (`SimplexWater` Optimality). *For any* $\lambda \in \left[\frac{1}{2}, 1\right)$ *the maximum detection gap upper bound is attained by* ***SimplexWater****.*

Beyond Binary Scores

- Binary detection gap is capped by $\frac{m(1-\lambda)}{2(m-1)}$
- Bigger field size?

Beyond Binary Scores

- Binary detection gap is capped by $\frac{m(1-\lambda)}{2(m-1)}$
- Bigger field size?

Let's make f real!

Beyond Binary Scores

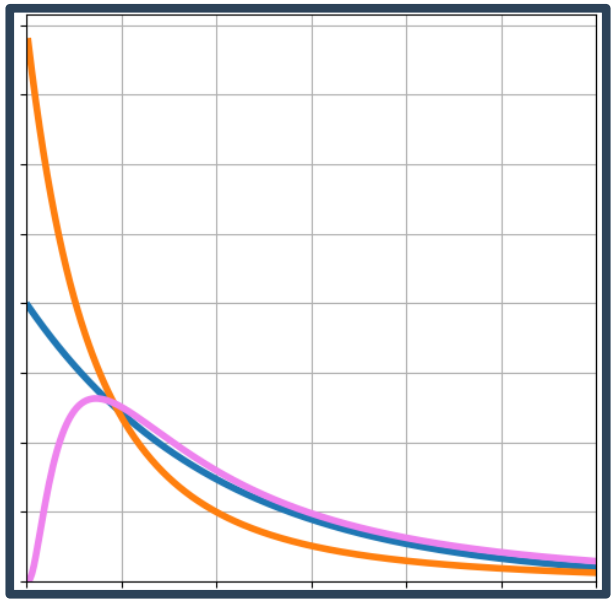
- Binary detection gap is capped by $\frac{m(1-\lambda)}{2(m-1)}$
- Bigger field size?

Let's make f real!

*(Leibniz)

HeavyWater

Continuous Score Functions

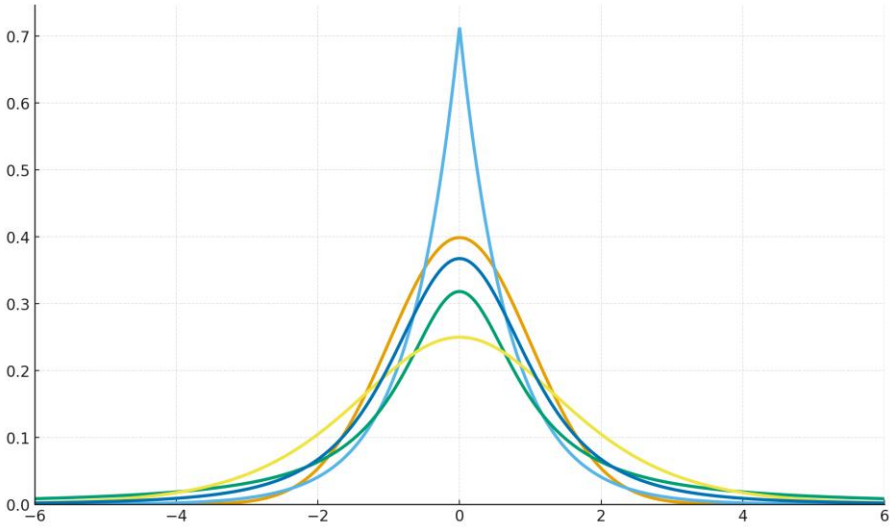


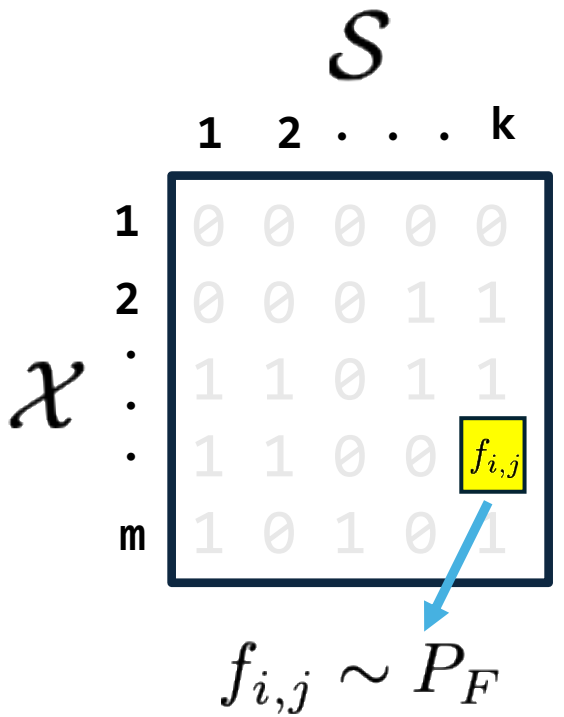
Towards a Continuous Watermark

- Extending **continuous** scores – Sample f **randomly** P_F
  - Practically - Sample the lookup entries i.i.d. P_F

$$f \sim P_F$$

- Distribution P_F is **continuous, zero mean 1D**



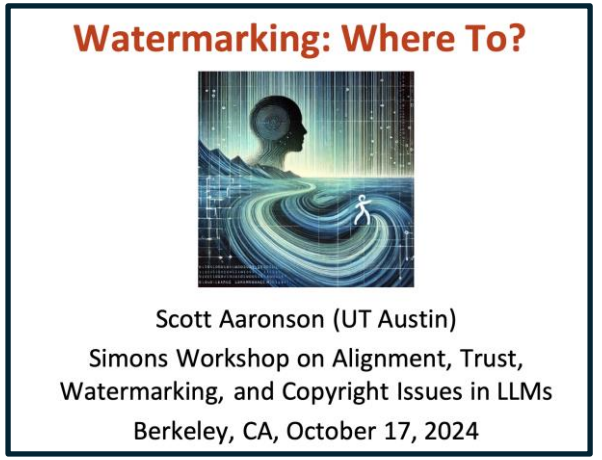


Random Score Generalizes Existing Watermarks

- **Gumbel** Watermark (Aaronson & Kircher ‘23, OpenAI):
  - Add i.i.d. Gumbel(0,1) noise to the logits and take argmax:

$$x = \arg\max_{i \in [1:m]} \ell_i + g_i \quad \text{(no reweigh)}$$

  - Detection:
    - Resample Gumbel variables and calculate a Gumbel-based statistic



Random Score Generalizes Existing Watermarks

- **Gumbel** Watermark (Aaronson & Kircher ‘23, OpenAI):
  - Add i.i.d. Gumbel(0,1) noise to the logits and take argmax:

$$x = \arg\max_{i \in [1:m]} \ell_i + g_i \quad \text{(no reweigh)}$$

  - Detection:
    - Resample Gumbel variables and calculate a Gumbel-based statistic

- Gumbel watermark is asymptotically a **special case** of our framework!

Theorem 3 (Gumbel Watermark as OT). *When the score random variables $f(x, s)$, are sampled i.i.d. from Gumbel$(0, 1)$, the solution to the OT problem converges to the Gumbel watermark as $|\mathcal{S}| = k \to \infty$.*

Random Score Generalizes Existing Watermarks

- **Gumb**
  - Ad
  - De
    -

If Gumbel is a special case
Can we provide a better solution?

- Gumbel watermark is asymptotically a **special case** of our framework!

Theorem 3 (Gumbel Watermark as OT). *When the score random variables $f(x, s)$, are sampled i.i.d. from Gumbel$(0, 1)$, the solution to the OT problem converges to the Gumbel watermark as $|\mathcal{S}| = k \to \infty$.*

Towards **`HeavyWater`**

- What controls the detection gap?

Towards **HeavyWater**

- What controls the detection gap?
- Generally, detection depends on the **tail of** P_F !

Theorem 4 (Detection Gap). *Let* $\lambda \in \left[\frac{1}{2}, 1\right)$, *and consider the score difference random variable* $\Delta = f(x,s) - f(x',s')$ *for some* $(x,s) \neq (x',s')$, *where* $f(x,s)$ *and* $f(x',s')$ *are sampled i.i.d. from* P_F. *Let the cumulative distribution function of* Δ *be* F, *and let* $Q = F^{-1}$ *be its inverse. Then,*

$$\lim_{k\to\infty} \mathsf{D}_{\mathsf{gap}}^{[P_F]}(m,k,\lambda) = \int_{1-\lambda}^{1} Q(u)du, \tag{8}$$

Towards **HeavyWater**

- What controls the detection gap?
- Generally, detection depends on the **tail of** P_F !

Theorem 4 (Detection Gap). *Let* $\lambda \in \left[\frac{1}{2}, 1\right)$, *and consider the score difference random variable* $\Delta = f(x,s) - f(x',s')$ *for some* $(x,s) \neq (x',s')$, *where* $f(x,s)$ *and* $f(x',s')$ *are sampled i.i.d. from* P_F. *Let the cumulative distribution function of* Δ *be* F, *and let* $Q = F^{-1}$ *be its inverse. Then,*

$$\lim_{k\to\infty} \mathsf{D}_{\mathsf{gap}}^{[P_F]}(m,k,\lambda) = \int_{1-\lambda}^{1} Q(u)du, \tag{8}$$

- Define score difference Δ
- Quantile of the distribution of Δ is Q
- Detection gap is control by the integral of Q

Towards **HeavyWater**

Theorem 4 (Detection Gap). *Let $\lambda \in \left[\frac{1}{2}, 1\right)$, and consider the score difference random variable $\Delta = f(x,s) - f(x',s')$ for some $(x,s) \neq (x',s')$, where $f(x,s)$ and $f(x',s')$ are sampled i.i.d. from P_F. Let the cumulative distribution function of Δ be F, and let $Q = F^{-1}$ be its inverse. Then,*

$$\lim_{k\to\infty} \mathsf{D}_{\mathsf{gap}}^{[P_F]}(m,k,\lambda) = \int_{1-\lambda}^{1} Q(u)du, \tag{8}$$

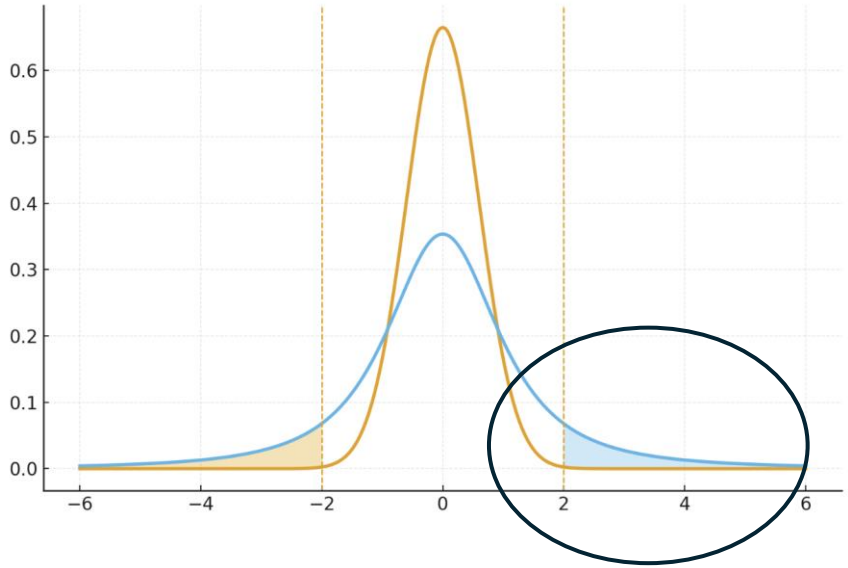
- Heavier tail Δ => Bigger detection gap

Towards **HeavyWater**

Theorem 4 (Detection Gap). *Let $\lambda \in \left[\frac{1}{2}, 1\right)$, and consider the score difference random variable $\Delta = f(x,s) - f(x',s')$ for some $(x,s) \neq (x',s')$, where $f(x,s)$ and $f(x',s')$ are sampled i.i.d. from P_F. Let the cumulative distribution function of Δ be F, and let $Q = F^{-1}$ be its inverse. Then,*

$$\lim_{k\to\infty} \mathsf{D}_{\mathsf{gap}}^{[P_F]}(m,k,\lambda) = \int_{1-\lambda}^{1} Q(u)du, \tag{8}$$

- Heavier tail Δ => Larger detection gap
- Heavy tailed f => Heavy tailed Δ
  *(through MGFs)



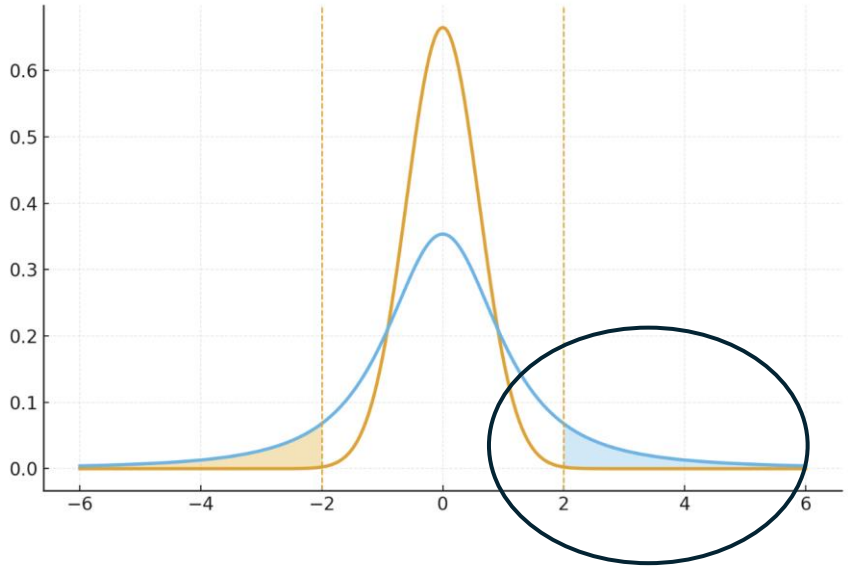
Towards **HeavyWater**

Theorem 4 (Detection Gap). *Let $\lambda \in \left[\frac{1}{2}, 1\right)$, and consider the score difference random variable $\Delta = f(x,s) - f(x',s')$ for some $(x,s) \neq (x',s')$, where $f(x,s)$ and $f(x',s')$ are sampled i.i.d. from P_F. Let the cumulative distribution function of Δ be F, and let $Q = F^{-1}$ be its inverse. Then,*

$$\lim_{k\to\infty} \mathsf{D}_{\mathsf{gap}}^{[P_F]}(m,k,\lambda) = \int_{1-\lambda}^{1} Q(u)du, \tag{8}$$

- Heavier
- Heavy t
  *(throu

Sample score from a heavy tailed distribution!

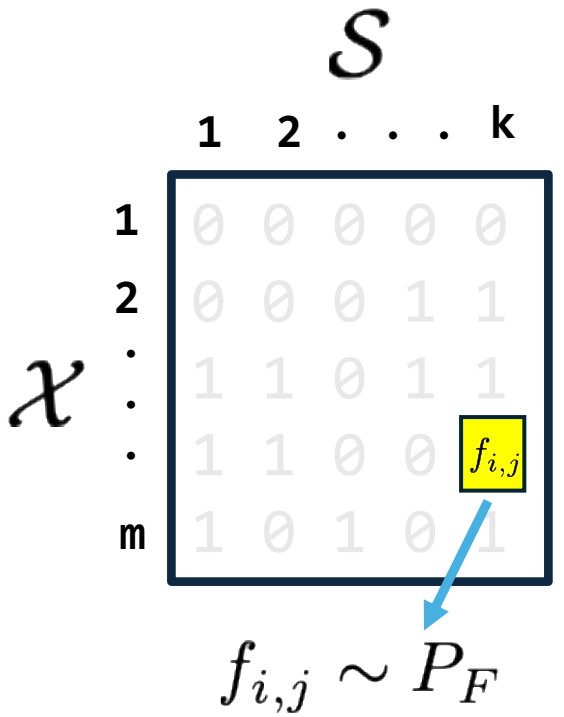


`HeavyWater`

- `HeavyWater` – Sample f from a heavy tailed distribution P_F

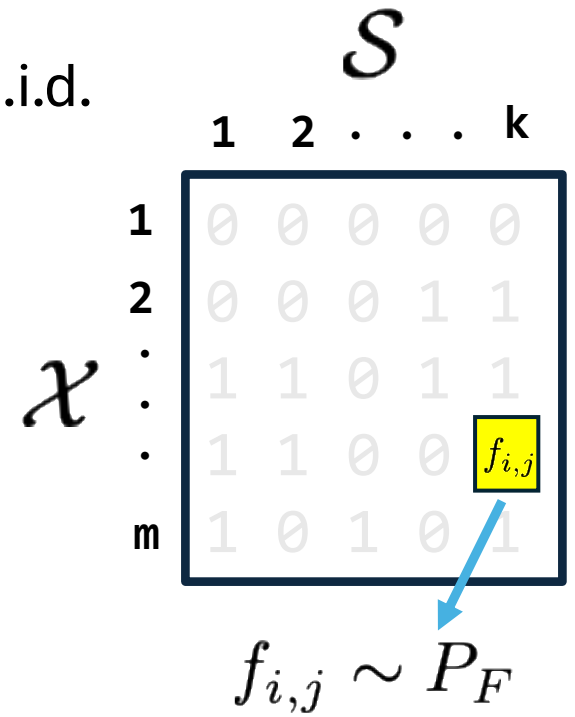
HeavyWater

- `HeavyWater` – Sample f from a heavy tailed distribution P_F
- Best empirical performance – Lognormal distribution



HeavyWater

- `HeavyWater` – Sample f from a heavy tailed distribution P_F
- Best empirical performance – Lognormal distribution
- `HeavyWater` Steps:
  - **Set** side information size k and sample a $m \times k$ lookup table i.i.d.
  - **Share** sampled lookup table prior to text generation
  - **Repeat** watermarking algorithm



Beyond Zero-Distortion

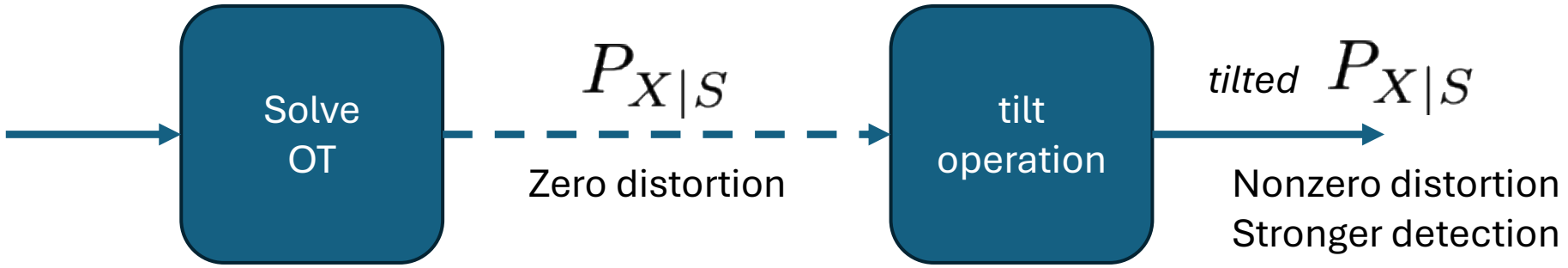
- Recall: $\mathbb{E}_{P_S}[P_{X|S}] = P_X$ (due to OT)

Beyond Zero-Distortion

- Recall: $\mathbb{E}_{P_S}[P_{X|S}] = P_X$ (due to OT)
- **Higher detectability** – at the cost of **inducing distortion**

Beyond Zero-Distortion

- Recall: $\mathbb{E}_{P_S}[P_{X|S}] = P_X$ (due to OT)
- **Higher detectability** – at the cost of **inducing distortion**
- pushing expected scores **further apart**
- Alter $P_{X|S}$ to **increase** expected score - tilting
  - Exact operation $\text{tilt}(P_{X|S})$ depends on exact structure of f



Beyond Zero-Distortion

- Tilting `SimplexWater`:
  - Increase probability of tokens with score 1
  - Decrease probability of tokens with score 0

$$\text{tilt}(P^*_{X|S=s}, s, \delta) = P^*_{X|S=s}(x, s)\left(1 + \delta \cdot (\mathbf{1}_{\{f(x,s)=1\}} - \mathbf{1}_{\{f(x,s)=0\}})\right)$$

- Tilting `HeavyWater` – around its (zero) mean

$$\text{tilt}(P^*_{X|S=s}, s, \delta) = P^*_{X|S=s}(x, s)\left(1 + \delta \cdot \text{sign}(f(x, s))\right)$$

Numerical Results

Setting

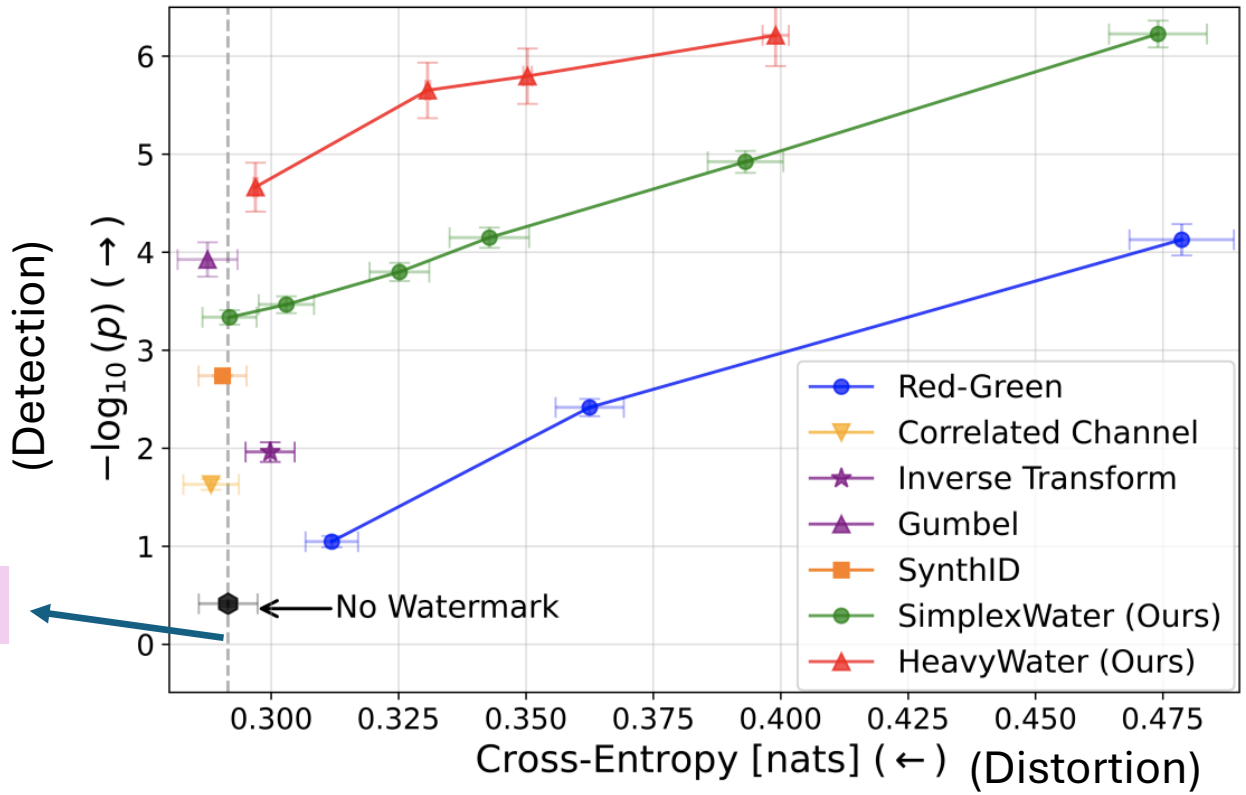
- LLMs – Llama2-7b, Llama3-8b, Mistral-v02-7b
- 7 Datasets, For example:
  - FinanceQA
  - LCC
- Top-p sampling, 0.99
- Temperature 1.0/ 0.7
- Methods: Red-Green, Gumbel, Inverse transform, Correlated Channel, SynthID
- Experiments from 2 benchmarks – WaterBench and MarkMyWords

Detection-Distortion Tradeoff

- **Detection**: p-value of statistical test
- **Distortion**: Cross entropy between base and watermarked distributions

Detection-Distortion Tradeoff

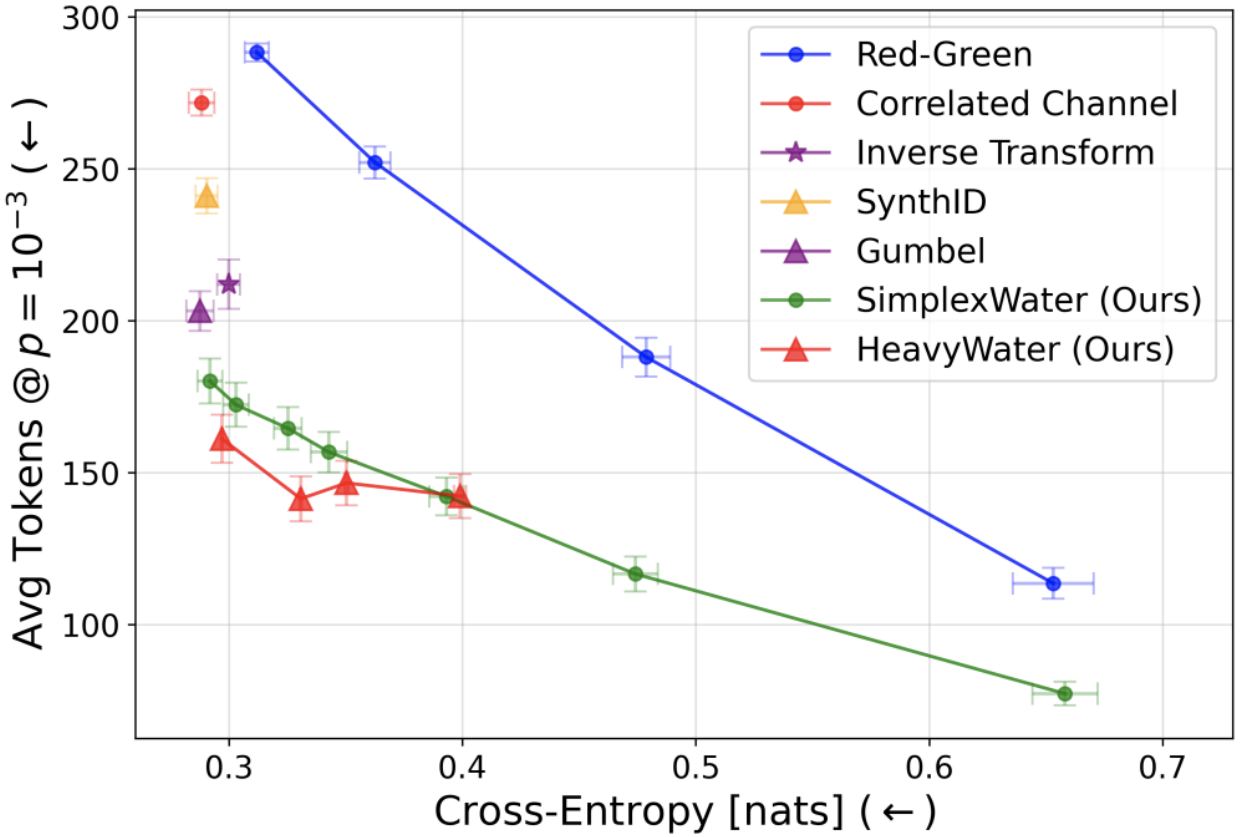
- **Detection**: p-value of statistical test
- **Distortion**: Cross entropy between base and watermarked distributions



Zero distortion

Watermark Size

- Watermark Size - #tokens at detection to obtain a certain p-value.



Randomness Generation

- Theoretical assumption - Perfect randomness – s is i.i.d.
  - New independent key on each time step
- In practice – The key is a function of previous tokens
  - Robustness
- Examples of dependence
  - Markov $k_t = f(k_0, x_{t-1})$
  - Product $k_t = k_0 \cdot \prod_{i=0}^{T} x_{t-i}$
  - Sum $k_t = k_0 + \sum_{i=0}^{T} x_{t-i}$

The Benefit of Switching

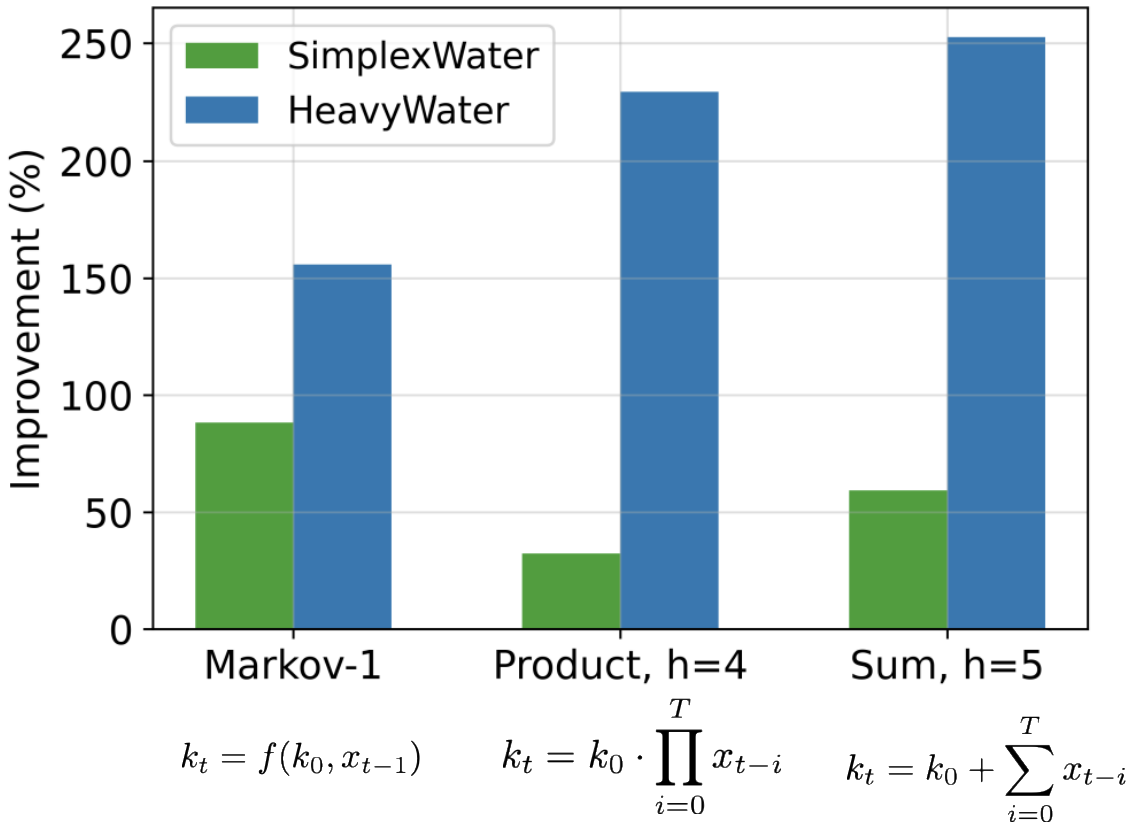
- Base method – Red-Green with $\delta = 2$ - nonzero distortion

The Benefit of Switching

- Base method – Red-Green with $\delta = 2$ - nonzero distortion
- Hashing schemes:
- Markov: $k_t = f(k_0, x_{t-1})$
- Prod: $k_t = k_0 \cdot \prod_{i=0}^{T} x_{t-i}$
- Sum: $k_t = k_0 + \sum_{i=0}^{T} x_{t-i}$

The Benefit of Switching

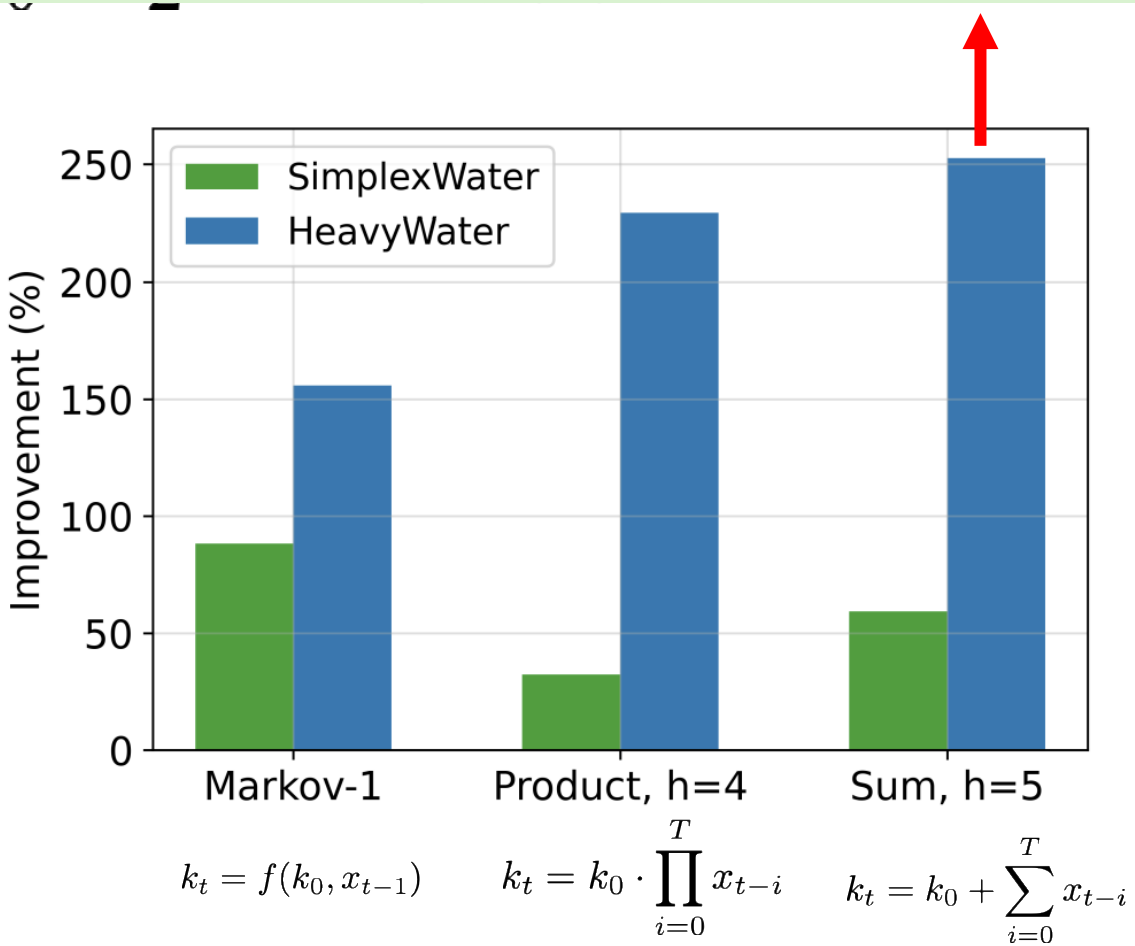
- Base method – Red-Green with $\delta = 2$ - nonzero distortion
- Hashing schemes:
- Markov: $k_t = f(k_0, x_{t-1})$
- Prod: $k_t = k_0 \cdot \prod_{i=0}^{T} x_{t-i}$
- Sum: $k_t = k_0 + \sum_{i=0}^{T} x_{t-i}$



The Benefit of Switching

- Base method – Red-Green wit
- Hashing schemes:
- Markov: $k_t = f(k_0, x_{t-1})$
- Prod: $k_t = k_0 \cdot \prod_{i=0}^{T} x_{t-i}$
- Sum: $k_t = k_0 + \sum_{i=0}^{T} x_{t-i}$

More than x2 better detection with lower distortion!



Watermark Information Size

- How many bits does it take to imprint the watermark?
  - m - Tokenizer vocabulary size
  - k - side information size (design choice)
  - F - floating point precision [bits]

- Information efficiency – low resource devices

Watermark Information Size

- How many bits does it take to imprint the watermark?
  - m - Tokenizer vocabulary size
  - k - side information size (design choice)
  - F - floating point precision [bits]

Watermark	**# Bits**
Red-Green	m
Inverse Transform	$m\mathrm{F}$
Gumbel	$m\mathrm{F}$
`SimplexWater` **(ours)**	$\log(m)$
`HeavyWater` **(ours)**	$\log(k)$

- Information efficiency – stealing the watermark (recover key)

Watermark Information Size

- How many bits does it take to imprint the watermark?
  - m - Tokenizer vocabulary size
  - k - side information size (design choice)
  - F - floating point precision [bits]

Watermark	# Bits
Red-Green	m
Inverse Transform	$m\mathrm{F}$
Gumbel	$m\mathrm{F}$
`SimplexWater` **(ours)**	$\log(m)$
`HeavyWater` **(ours)**	$\log(k)$

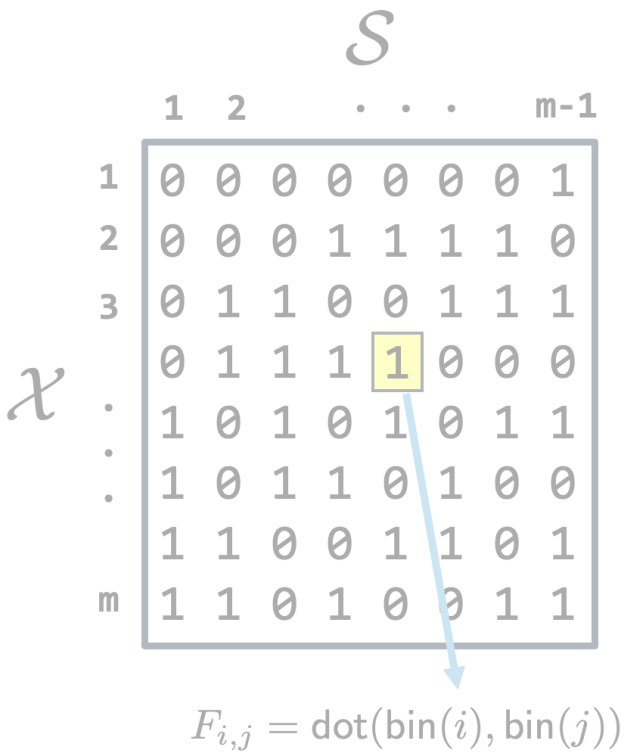
- Information efficiency – stealing the watermark (recover key)

Our watermarks are the most information-efficient!

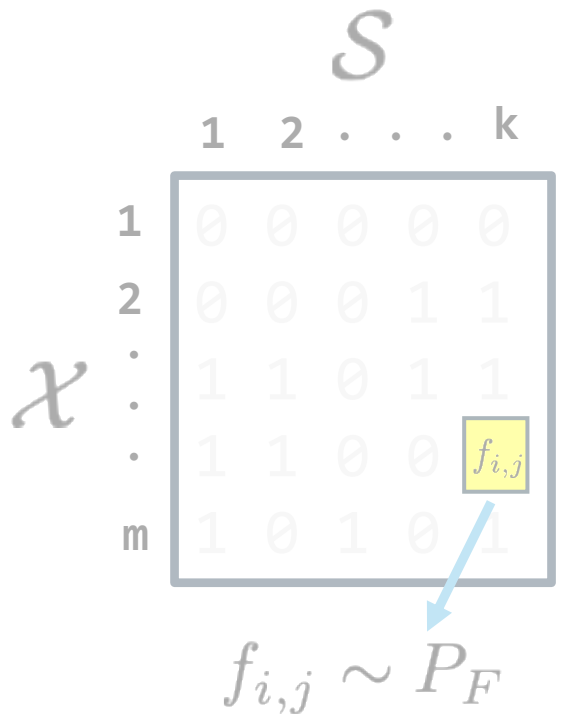
Conclusion

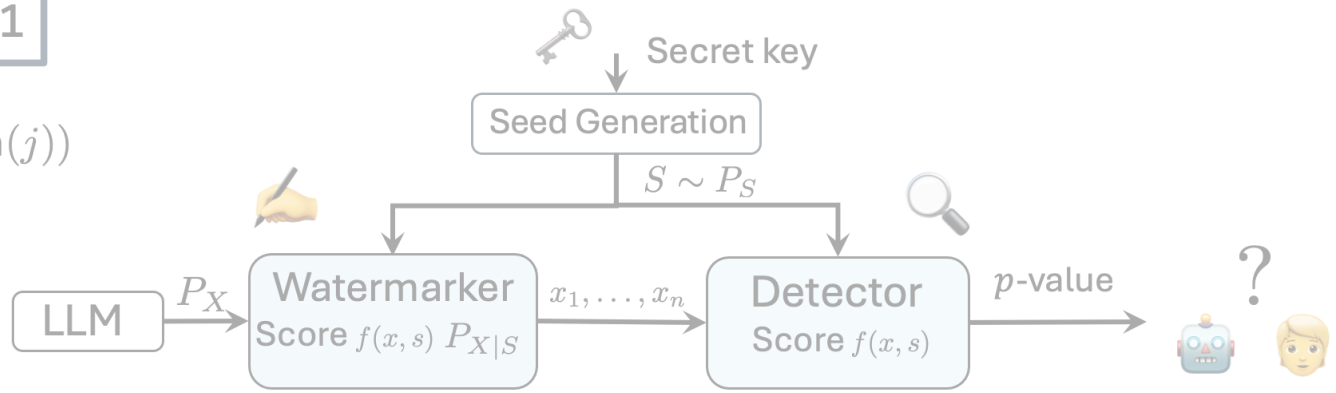
- Main Takeaways
  - Minmax optimization formulation $(f, P_X, P_{X|S})$
  - Optimal transport solution – zero-distortion watermark
  - Score design:
    - Binary scores – SimplexWater, Coding theory
    - Continuous scores – HeavyWater, Heavy-tailed distributions
  - First unifying framework – RG&Gumbel
  - Additional results – computational overhead, additional detection&quality metrics, coding.

- Future work
  - Learnable score
  - Integrating robustness into optimization problem



Thank you!





$$\max_{f \in \mathcal{F}} \min_{P_X \in P_\lambda} \max_{P_{X,S}} \mathbb{E}_{P_S P_{X|S}}[f(X,S)] - \mathbb{E}_{P_S P_X}[f(X,S)]$$

Randomness Generation

- Theoretical assumption - Perfect randomness – s is i.i.d.
  - Shared randomness independent across time

- In practice – keys are dependent
  - Usually through hashing previous tokens

  - Markov
  - Product Hash
  - Semantic Hash

- We empirically explore the effect of Hashing
- Our watermark works with any randomness generation