Breaking Verifiable Delay Functions in the Random Oracle Model

Ziyi Guan

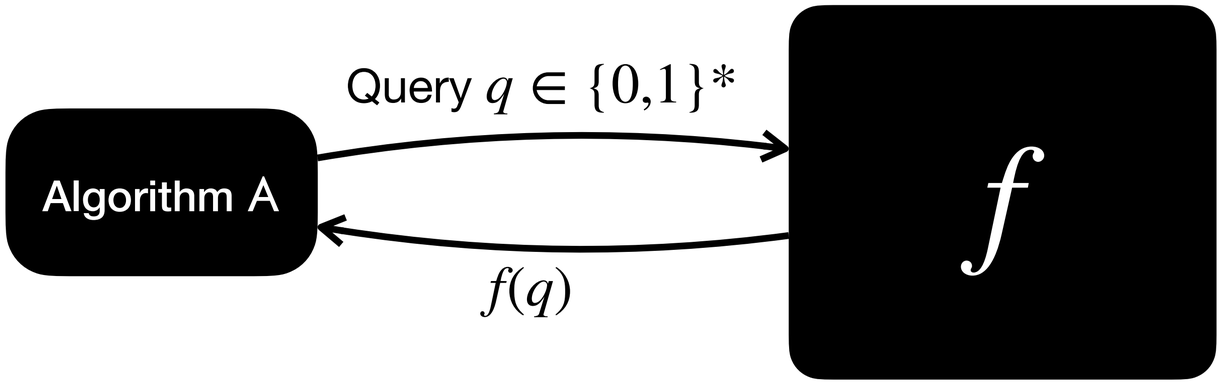
Joint work with Artur Riazanov, Weiqiang Yuan

EPFL

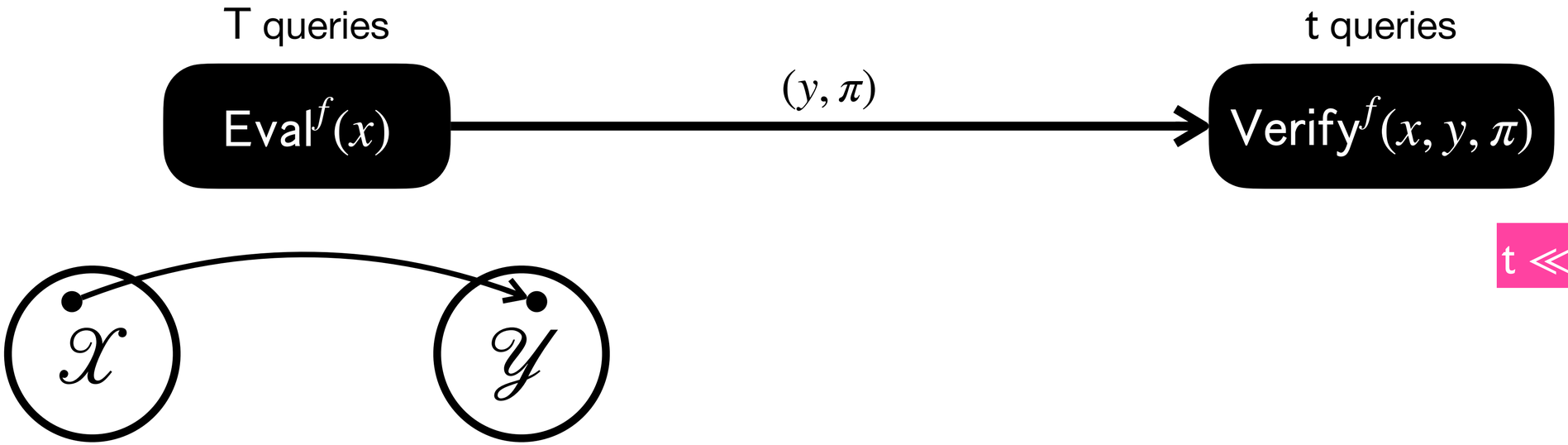
The random oracle model

Random oracle $\mathcal{O} := \{\mathcal{O}_\ell\}_{\ell \in \mathbb{N}}$

$\mathcal{O}_\ell$: uniform distribution over $f\colon \{0,1\}^* \to \{0,1\}^\ell$



Verifiable Delay Function (VDF)



$t \ll T$

FUNCTION → one **unique** output

DELAY

→ Can be evaluated in T queries

→ Cannot be evaluated in $o(T)$ rounds of queries

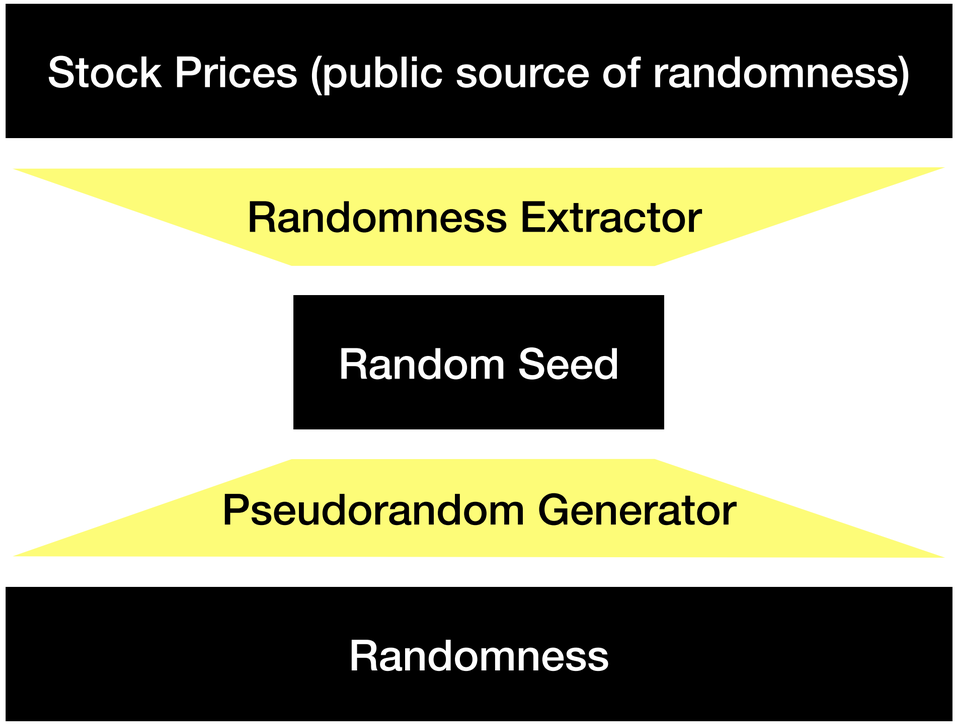
One can make multiple non-adaptive queries in one round

VERIFIABLE → correctness of output efficiently verifiable

Why study VDF?

Randomness beacon

- Publish randomness regularly
- Cannot predict/manipulate

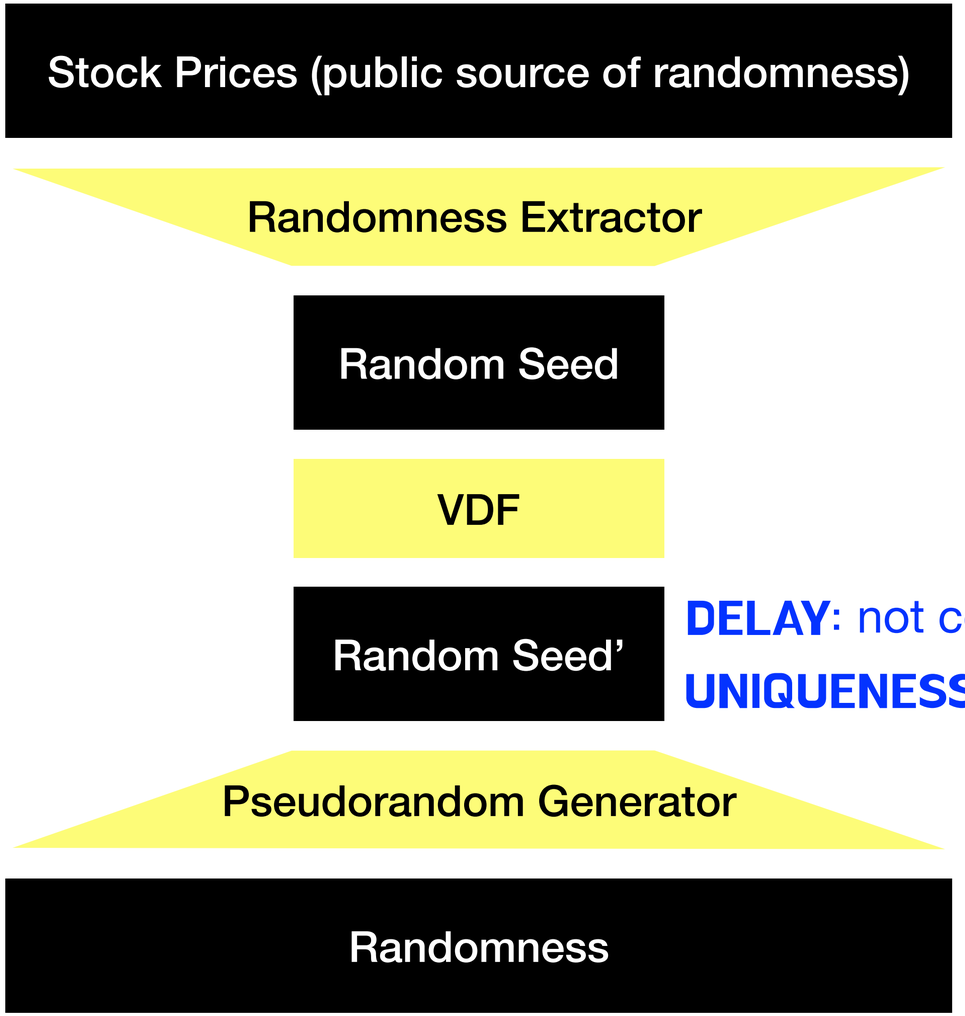


ISSUE: final randomness easy to compute & manipulate
(Stock prices can be biased/manipulated)

Why study VDF?

Randomness beacon

- Publish randomness regularly
- Cannot predict/manipulate

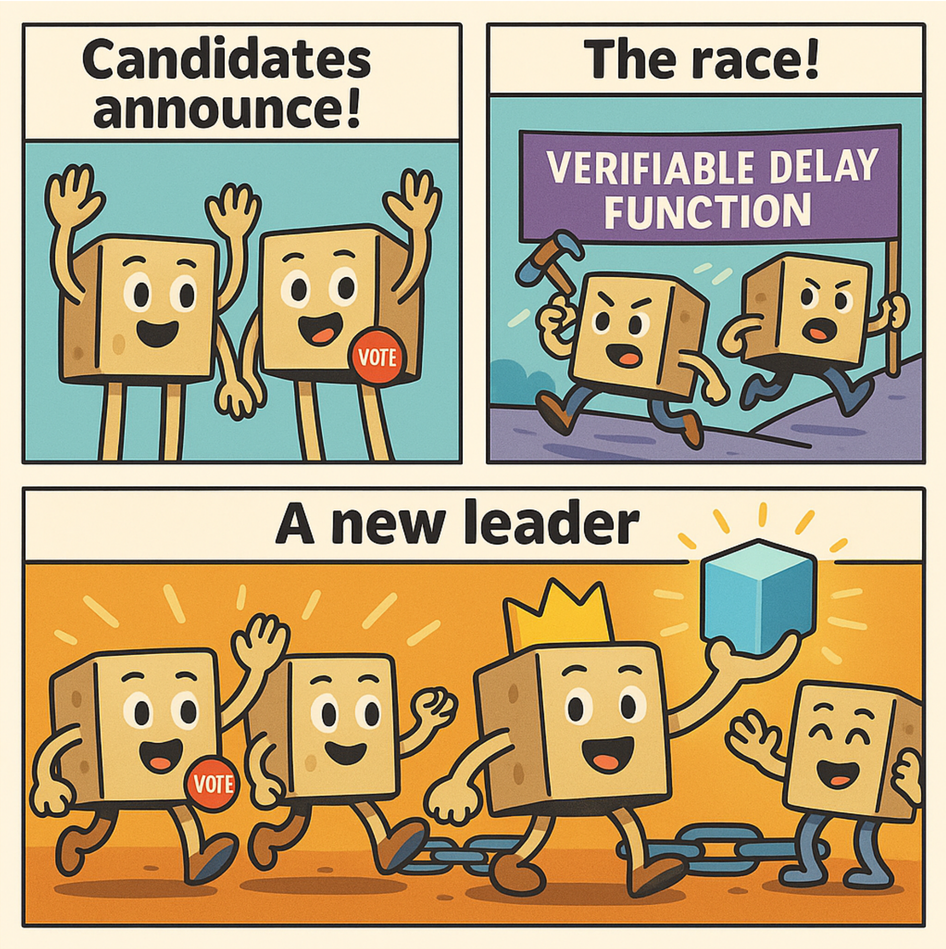


DELAY: not computable before market closes

UNIQUENESS: no ambiguity on output

Why study VDF?

Blockchain: leader election



UNIQUENESS → one **unique** leader

DELAY → cannot predict the next leader until shortly before the announcement

SOLANA Tezos chia IOTA

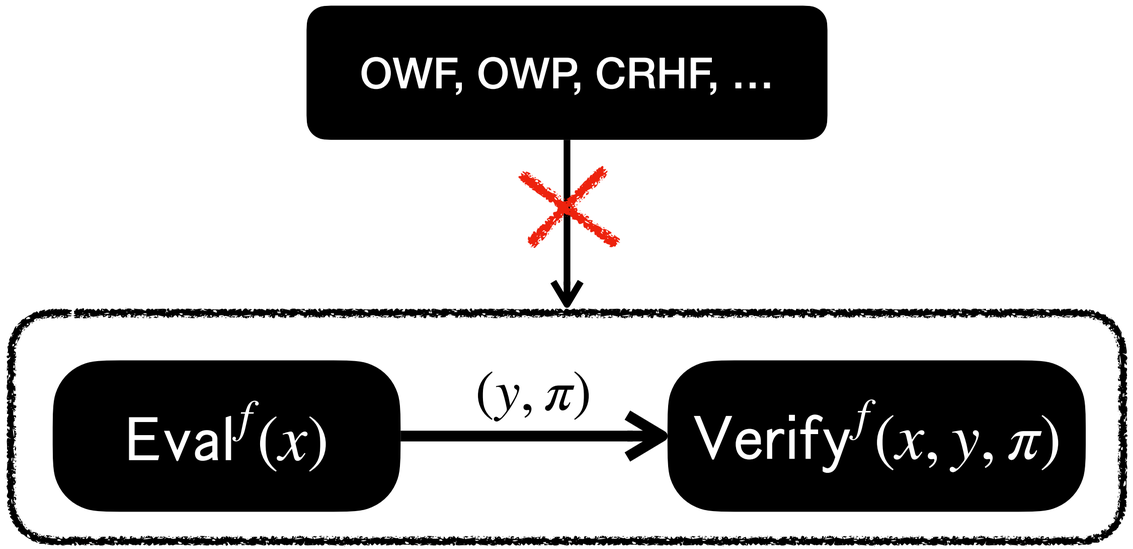
Verifiable Delay Functions Do Not Exist in the Random Oracle Model!!!!!!

But, VDFs exist in the standard model…?

Why do we care about ROM? It's not real anyway

What cryptography is needed for VDF constructions?

Existing VDFs rely on algebraic assumptions — not post-quantum secure



Complex assumptions (e.g. lattice) necessary for post-quantum VDF

But, VDFs exist in the standard model…?

Why do we care about ROM? It's not real anyway.

What security do VDF constructions have?

Standard model construction do not give concrete security analysis…
How to set security parameters in practice?

Incrementally Verifiable Computation
(Believed to not exist in the ROM)

Alternative idealized model (LDROM, AROM)

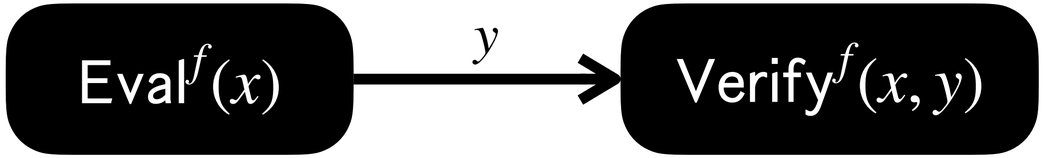
Non-succinct IVC in the ROM

…

Similar approach for VDFs?

Let’s get a little bit technical…

Query complexity vs. certificate complexity



Eval

G : $x \in \{0,1\}^n \mapsto b \in \{0,1\}$

Query complexity $D(G)$: min # bits in x needed to compute G

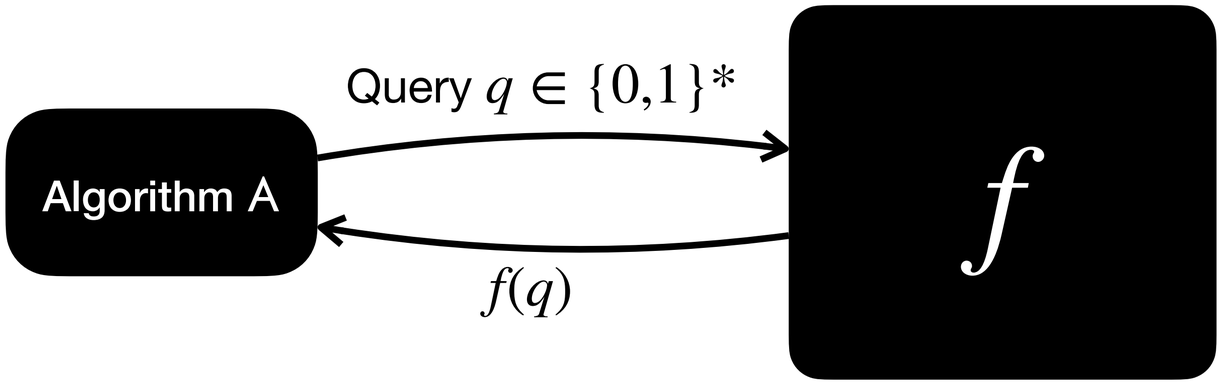
Certificate complexity $C(G)$: min # bits in x needed to prove $G(x) = 1$ or $G(x) = 0$

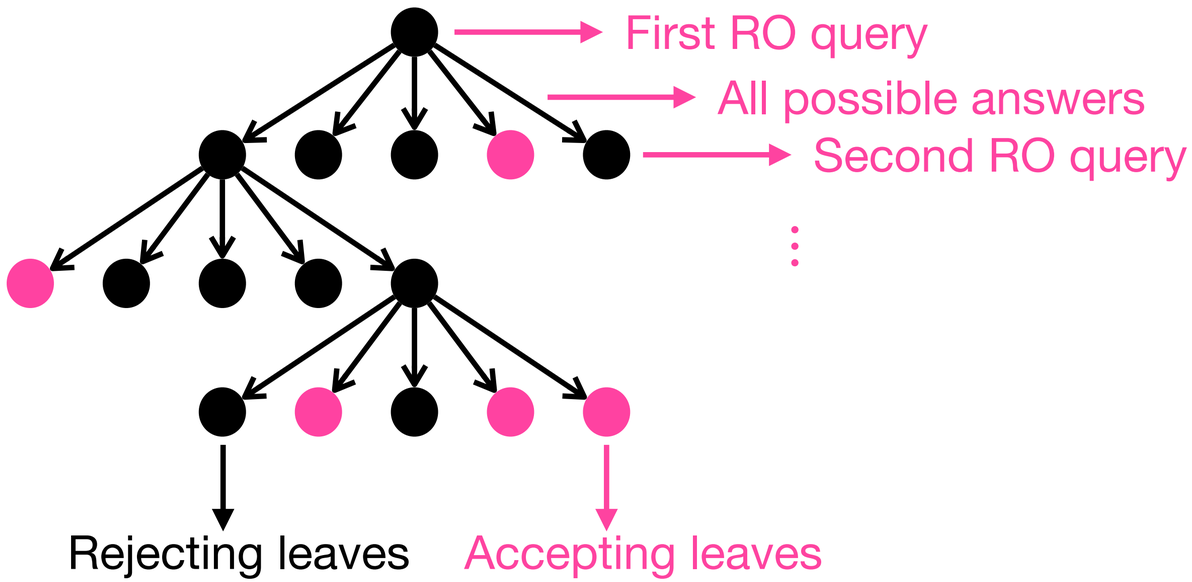
t (# queries made by $\mathsf{V}_{x,y}(f) := \text{Verify}^f(x, y)$)

$D(G) \leq C(G)^2$

$\Longrightarrow$ ∃ adv with at most t^2 queries that computes Eval
i.e. breaks the "DELAY" requirement

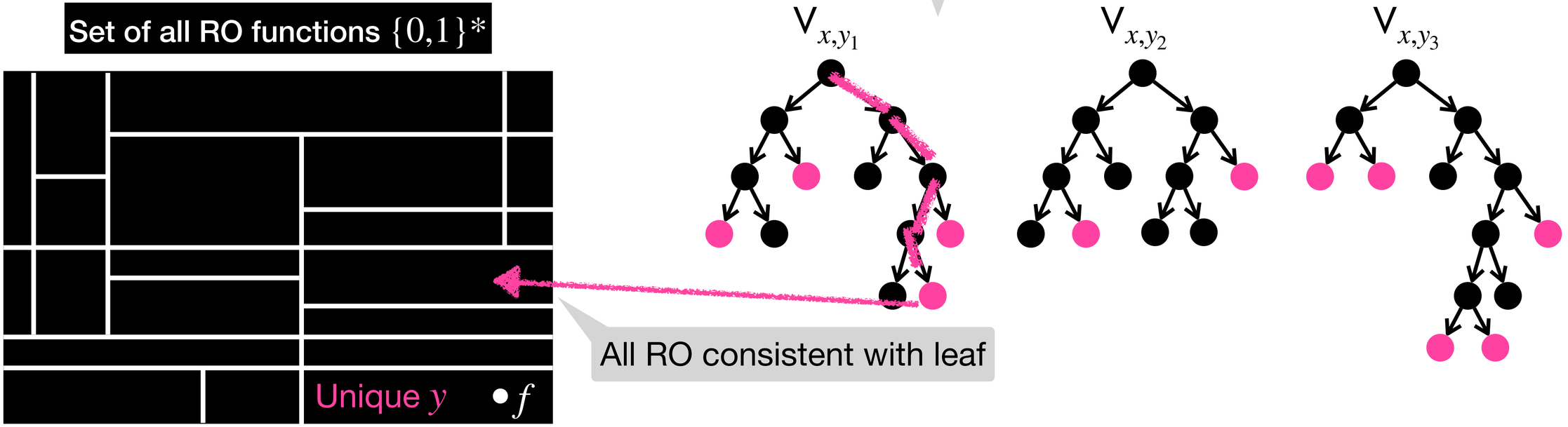
Decision tree algorithms in the ROM





Accepting leaves of $\mathsf{V}_{x,y}$ partitions random oracles

Leaf: a partial RO with at most t locations fixed ($\mathsf{Q}_{\mathsf{Verify}(f,x,y)}$)



(Correctness) $\forall f, \exists y$ (e.g. $y = \mathsf{Eval}^f(x)$) s.t. $\mathsf{Verify}^f(x, y) = 1$
(Uniqueness) $\forall f, \exists! y$ s.t. $\mathsf{Verify}^f(x, y) = 1$

i.e. the rectangles are disjoint!
$\Longrightarrow$ learning all queries in a rectangle can uniquely determine y

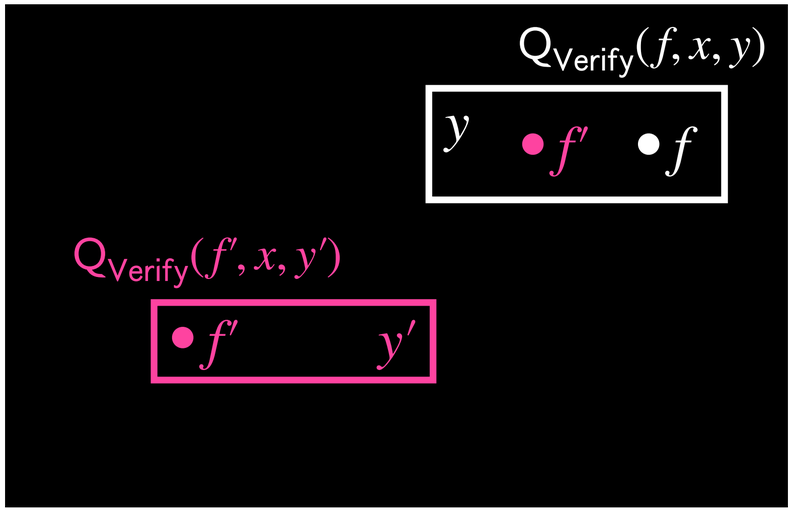
Our adversary computing Eval

$\mathsf{Adv}^{f}(x)$:

1. Initialize an empty evaluation table of f.
2. For $i \in [\mathrm{t}]$:
   a. Pick $f' \in \{0,1\}^*$ consistent with current f evaluation table.
   b. Compute $y' := \mathsf{Eval}^{f'}(x)$.
   c. Query f with $\mathsf{Q}_{\mathsf{Verify}}(f', x, y')$ in one round.
3. Output $y := \mathsf{Eval}^{f^\star}(x)$, where $f^\star$ is the current evaluation table of f.

t rounds of queries
At most t queries each round

No queries here!



Adv learns at least one query in $\mathsf{Q}_{\mathsf{Verify}}(f, x, y)$:

- $y' = y$: amazing!
- $y' \neq y$: $\exists q \in \mathsf{Q}_{\mathsf{Verify}}(f', x, y') \cap \mathsf{Q}_{\mathsf{Verify}}(f, x, y), f'[q] \neq f[q]$

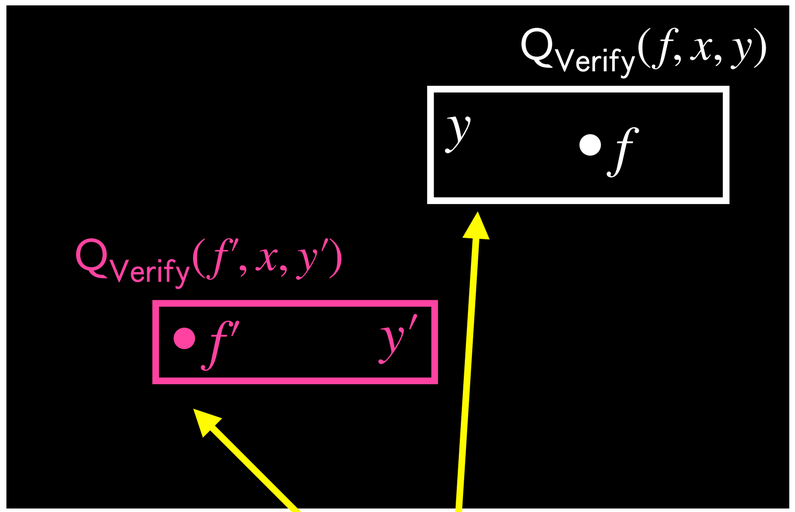
(Otherwise, $\mathsf{Verify}^{f}(x, y') = \mathsf{Verify}^{f'}(x, y') = 1$, breaking uniqueness)

Adv works only for "perfect VDF"!
"Computational VDF" needs extra tools…

Intersecting rectangles are problematic

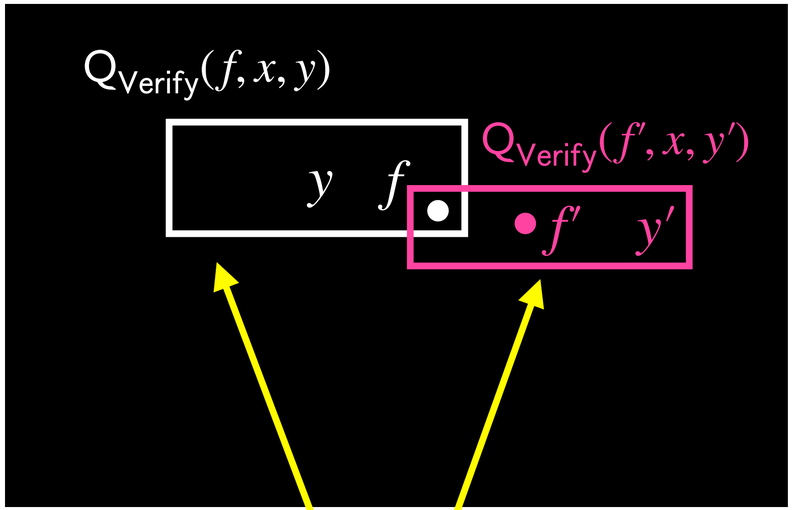
Perfect uniqueness: $\forall f,\ \exists!\ y$ s.t. $\mathsf{Verify}^f(x, y) = 1$
Computational uniqueness: $\mathrm{poly}(T)$-query adv **cannot** find $y' \neq \mathsf{Eval}^f(x)$
for randomly sampled f with high probability

Perfect uniqueness



$y \neq y' \Longrightarrow$ disjoint: can learn new location

Computationaly uniqueness



$y \neq y' \not\Longrightarrow$ disjoint: cannot learn new location

Need more advanced tools…

Recap

Verifiable Delay Functions Do Not Exist in the Random Oracle Model!!!!!!

i.e. “Uniqueness” and “Delay” are not compatible in the ROM

Open question:

- “Delay” alone is fine (proof of sequential work)
- How about “uniqueness” alone?

Thank you!