Breaking Verifiable Delay Functions in the Random Oracle Model

Great Random Oracle, bless me a puzzle that takes ages to solve, one glance to check, and exactly one right answer!

Ziyi Guan

Joint work with Artur Riazanov, Weiqiang Yuan

EPFL

The random oracle model

The random oracle model

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

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

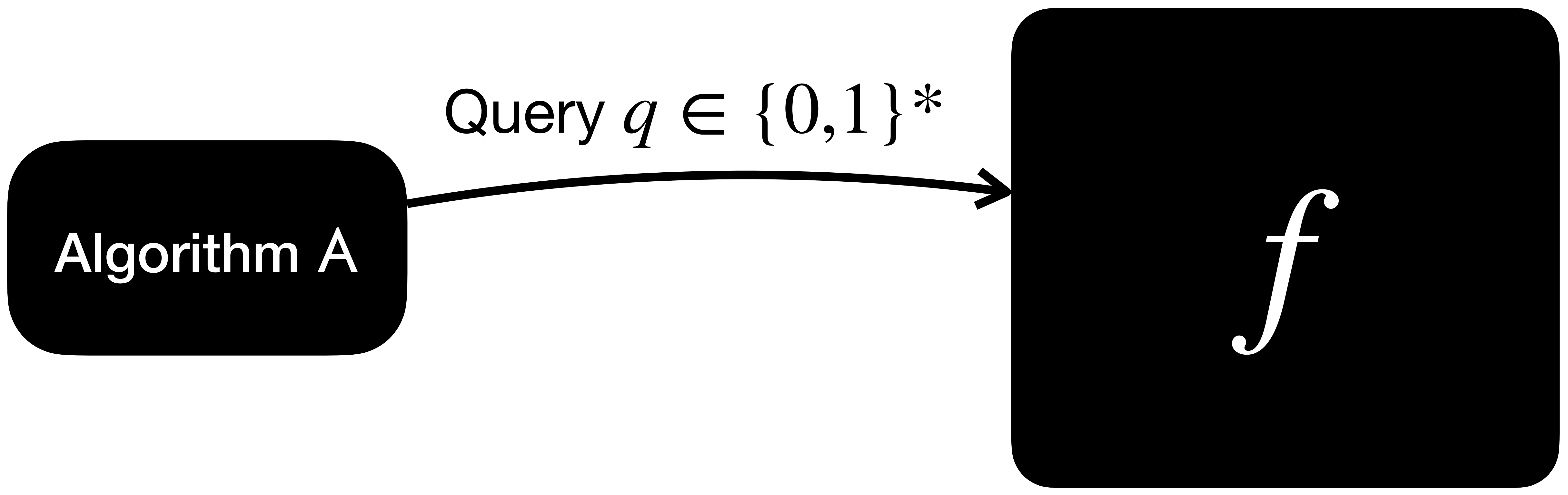
The random oracle model

Random oracle $\mathcal{O} := \{\mathcal{O}_\ell\}_{\ell \in \mathbb{N}}$
$\mathcal{O}_\ell$: uniform distribution over $f\colon \{0,1\}^* \to \{0,1\}^\ell$

Algorithm A
f

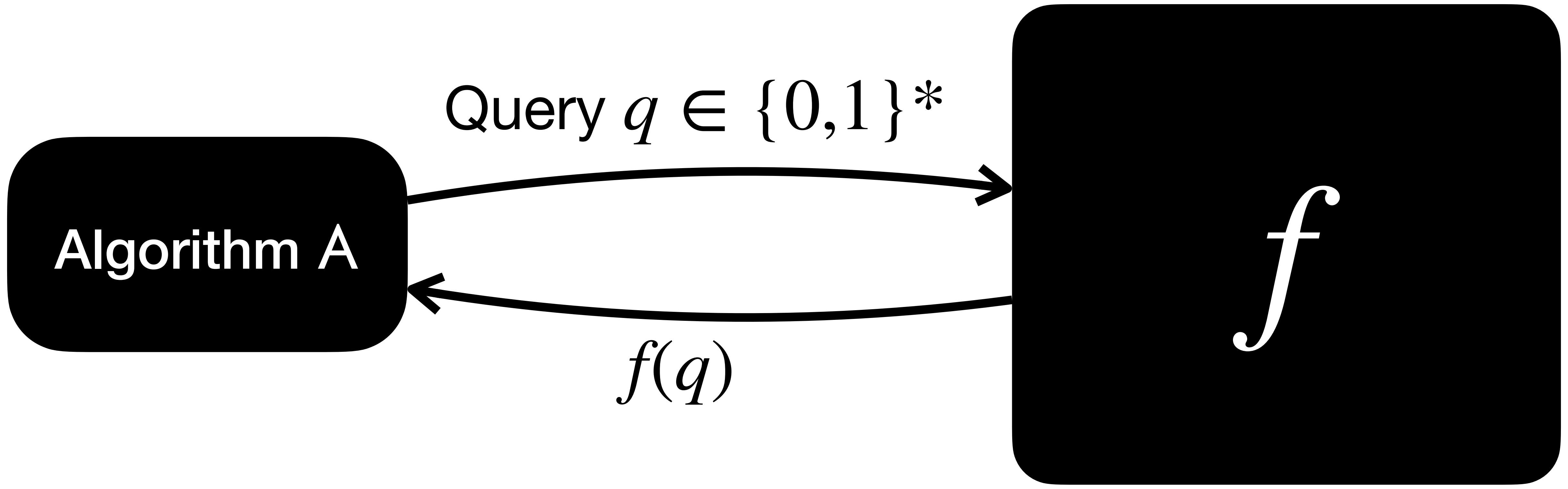
The random oracle model

Random oracle $\mathcal{O} := \{\mathcal{O}_\ell\}_{\ell \in \mathbb{N}}$
$\mathcal{O}_\ell$: uniform distribution over $f\colon \{0,1\}^* \to \{0,1\}^\ell$



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)

Verifiable Delay Function (VDF)

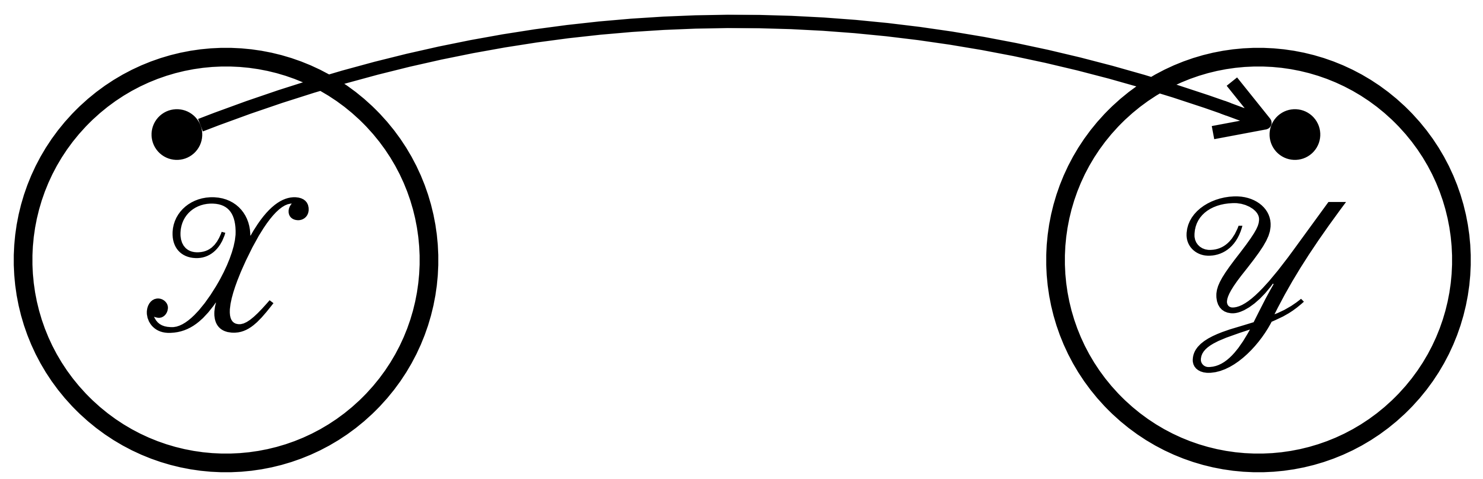
$\mathsf{Eval}^f(x)$

$\mathsf{Verify}^f(x, y, \pi)$

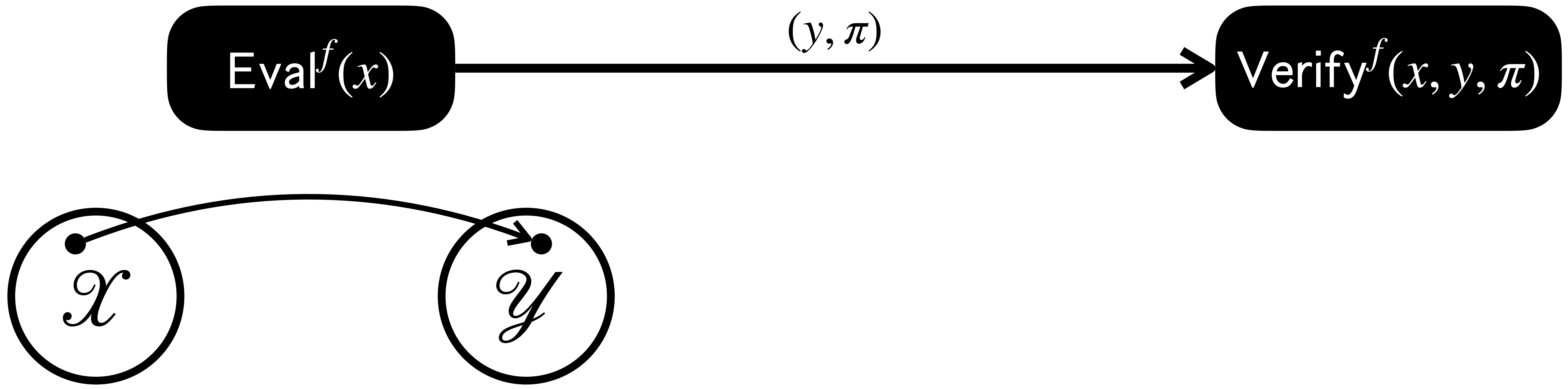
Verifiable Delay Function (VDF)

$\mathsf{Eval}^f(x)$

$\mathsf{Verify}^f(x, y, \pi)$

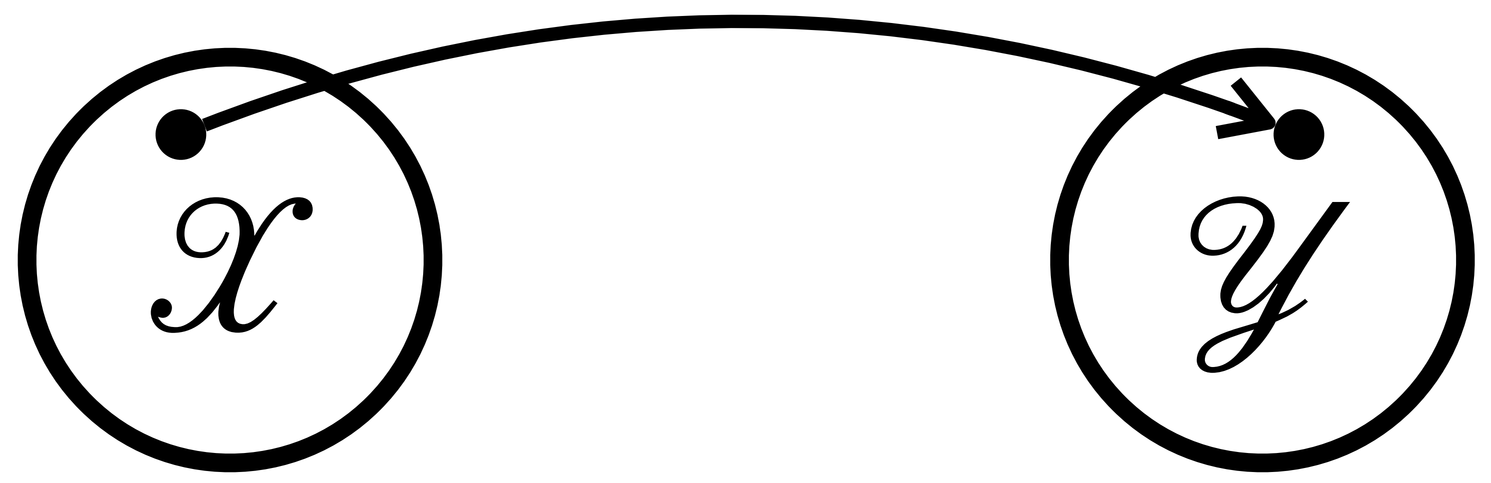


Verifiable Delay Function (VDF)



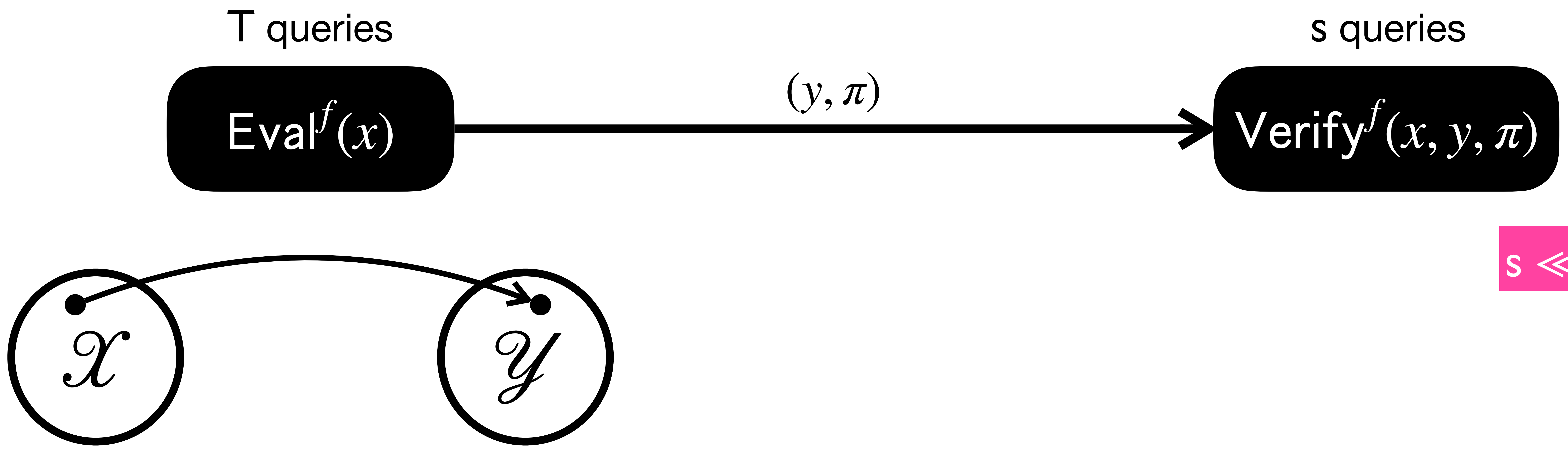
Verifiable Delay Function (VDF)

$$\boxed{\mathsf{Eval}^f(x)} \xrightarrow{(y,\pi)} \boxed{\mathsf{Verify}^f(x,y,\pi)}$$



VERIFIABLE → correctness of output efficiently publicly verifiable

Verifiable Delay Function (VDF)



$s \ll T$

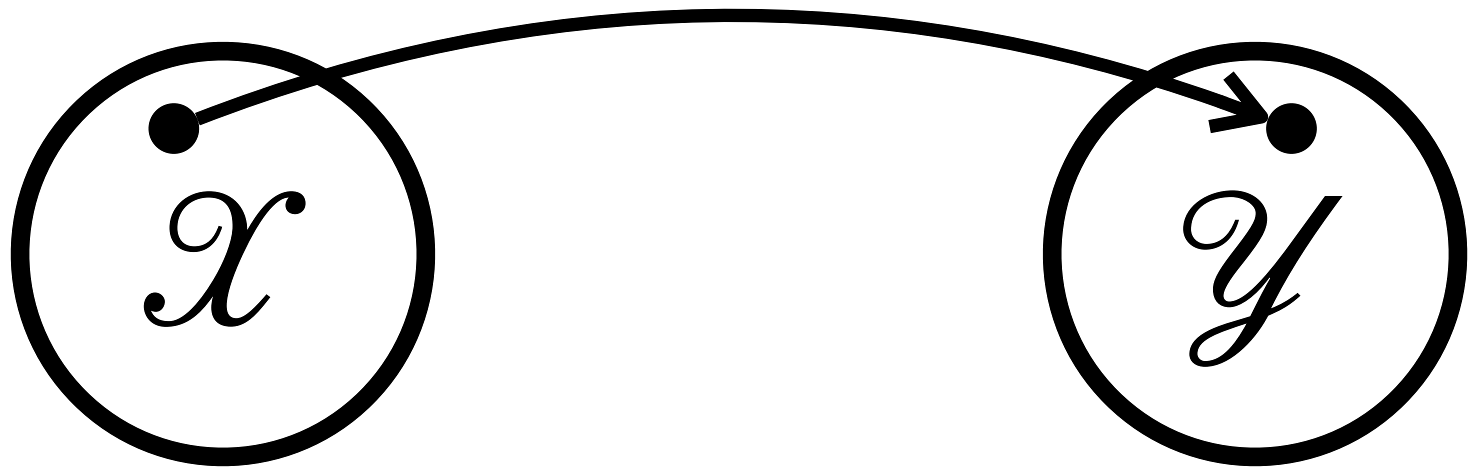
VERIFIABLE → correctness of output efficiently publicly verifiable

Verifiable Delay Function (VDF)

T queries s queries

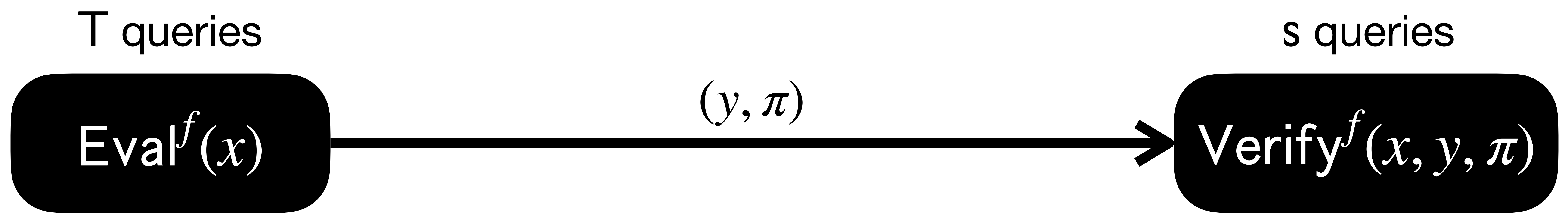
$$\begin{tikzcd}
\boxed{\mathsf{Eval}^f(x)} \arrow[r, "{(y, \pi)}"] & \boxed{\mathsf{Verify}^f(x, y, \pi)}
\end{tikzcd}$$

s $\ll$ T

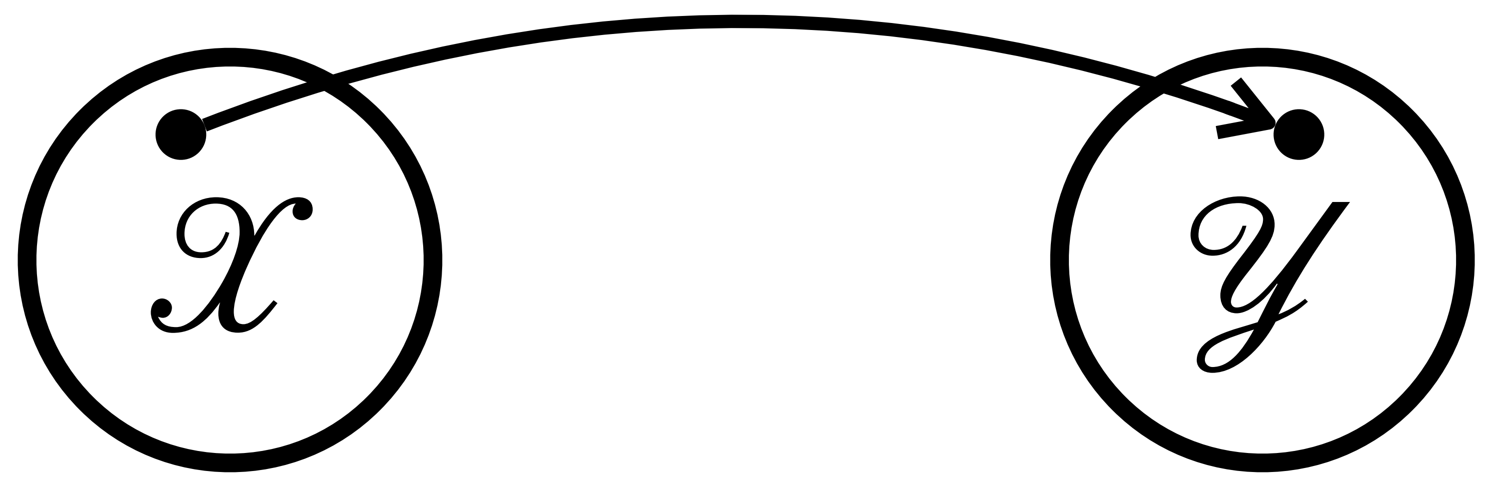


VERIFIABLE → correctness of output efficiently publicly verifiable
DELAY

Verifiable Delay Function (VDF)



s ≪ T



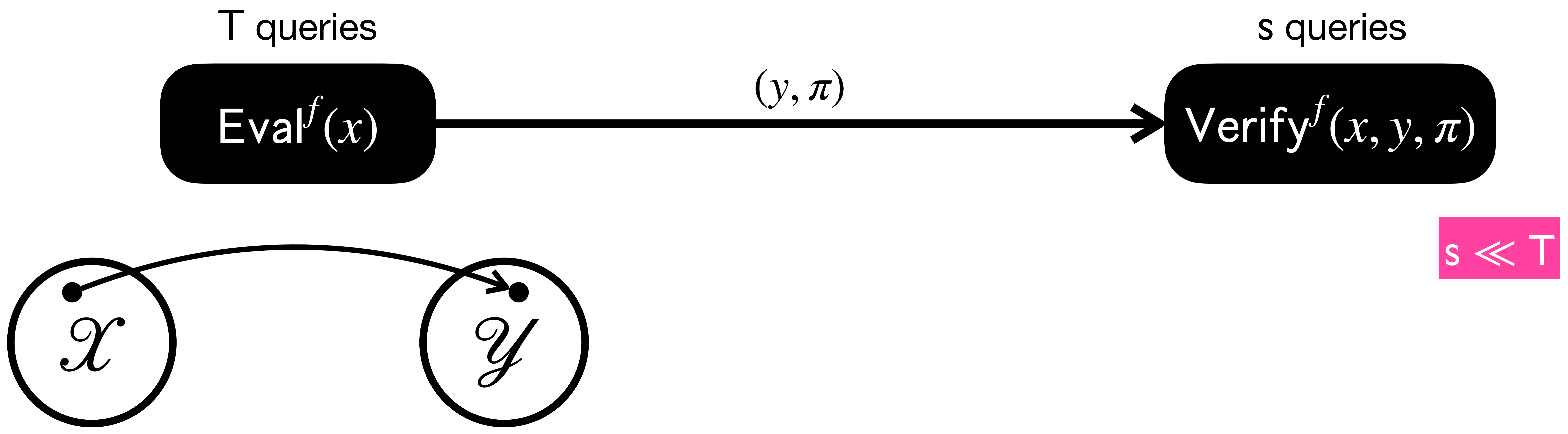
VERIFIABLE → correctness of output efficiently publicly verifiable

DELAY

→ Can be evaluated in T queries

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

Verifiable Delay Function (VDF)



VERIFIABLE → correctness of output efficiently publicly verifiable

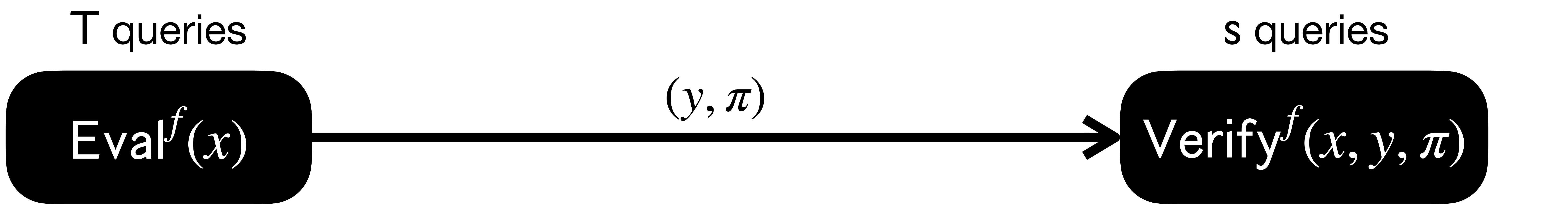
DELAY

→ Can be evaluated in T queries

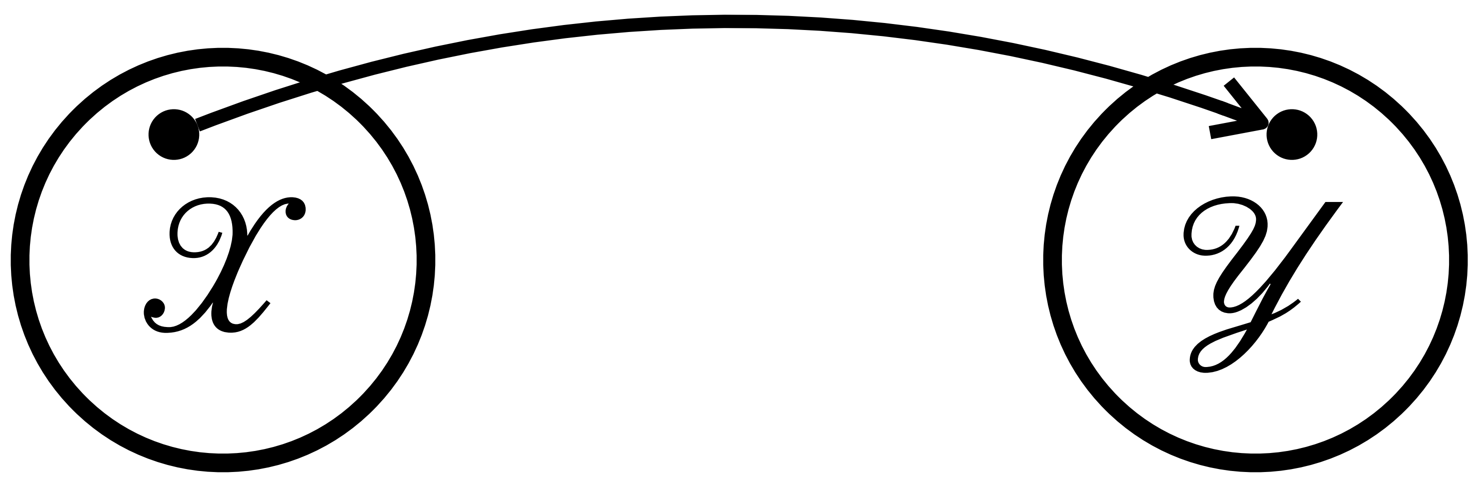
→ Cannot be evaluated in $o(T)$ rounds of queries

One can make multiple non-adaptive queries in one round

Verifiable Delay Function (VDF)



s ≪ T



VERIFIABLE → correctness of output efficiently publicly verifiable

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

FUNCTION → one **unique** output

Why study VDF?

Randomness beacon

Why study VDF?

Randomness beacon

- Publish randomness regularly
- Cannot predict/manipulate

Why study VDF?

Randomness beacon

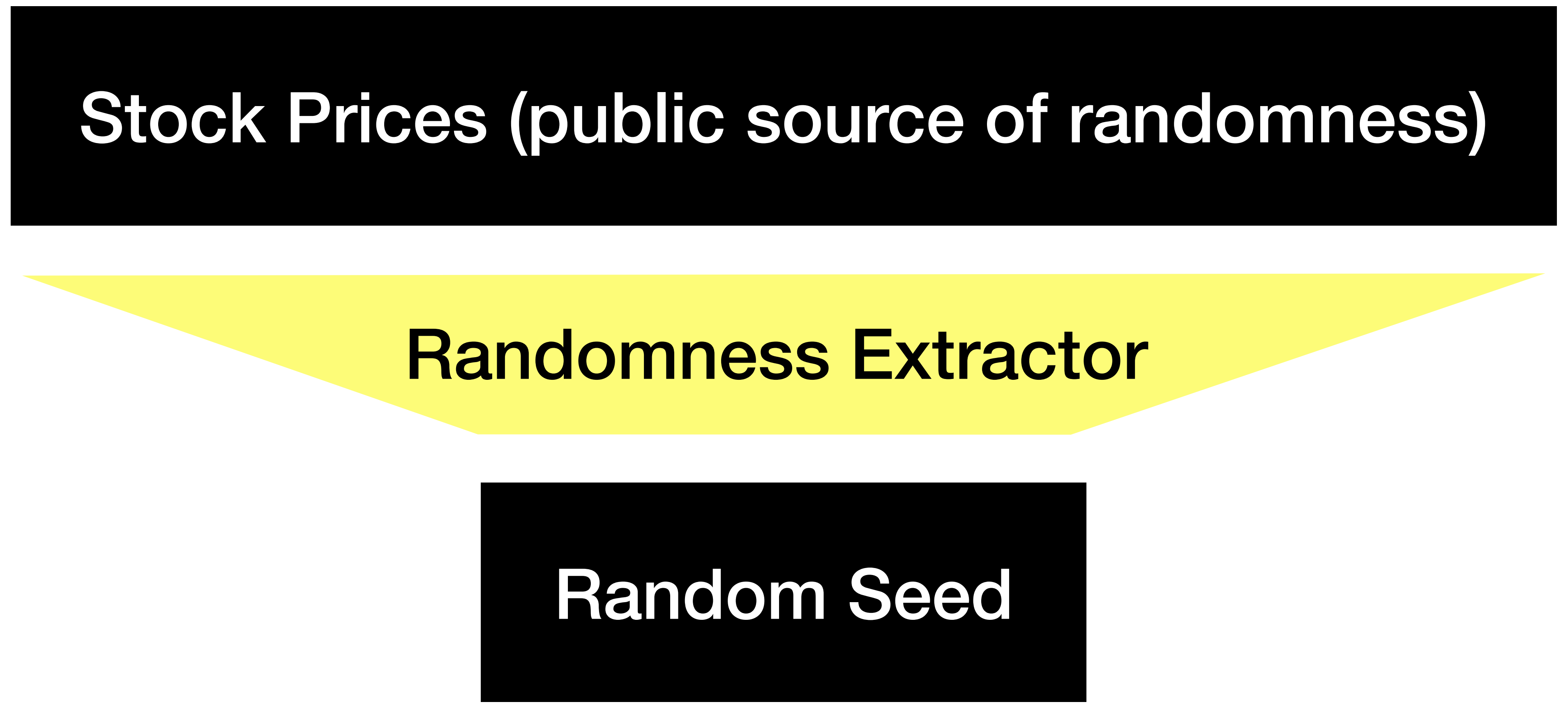
- Publish randomness regularly
- Cannot predict/manipulate

Stock Prices (public source of randomness)

Why study VDF?

Randomness beacon

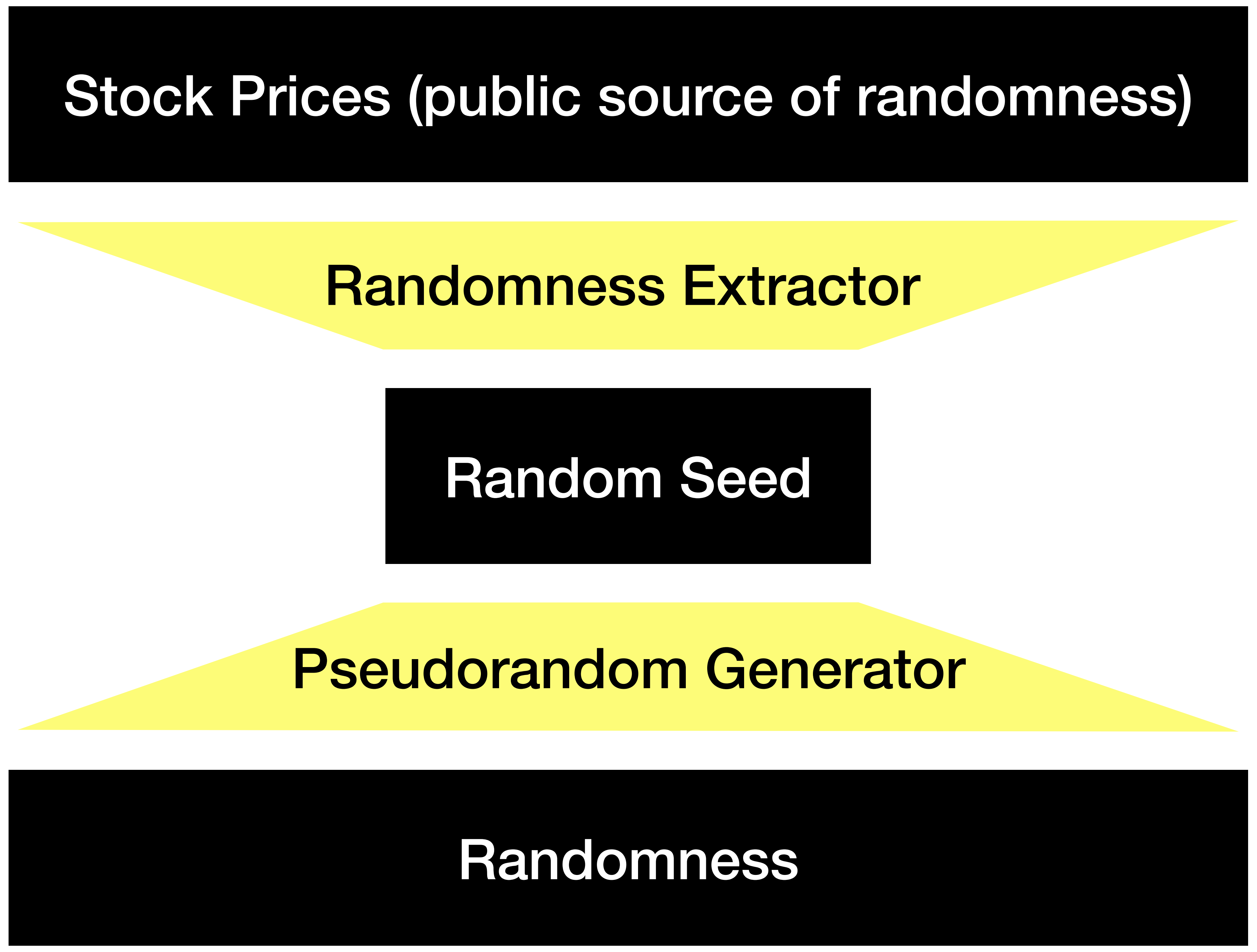
- Publish randomness regularly
- Cannot predict/manipulate



Why study VDF?

Randomness beacon

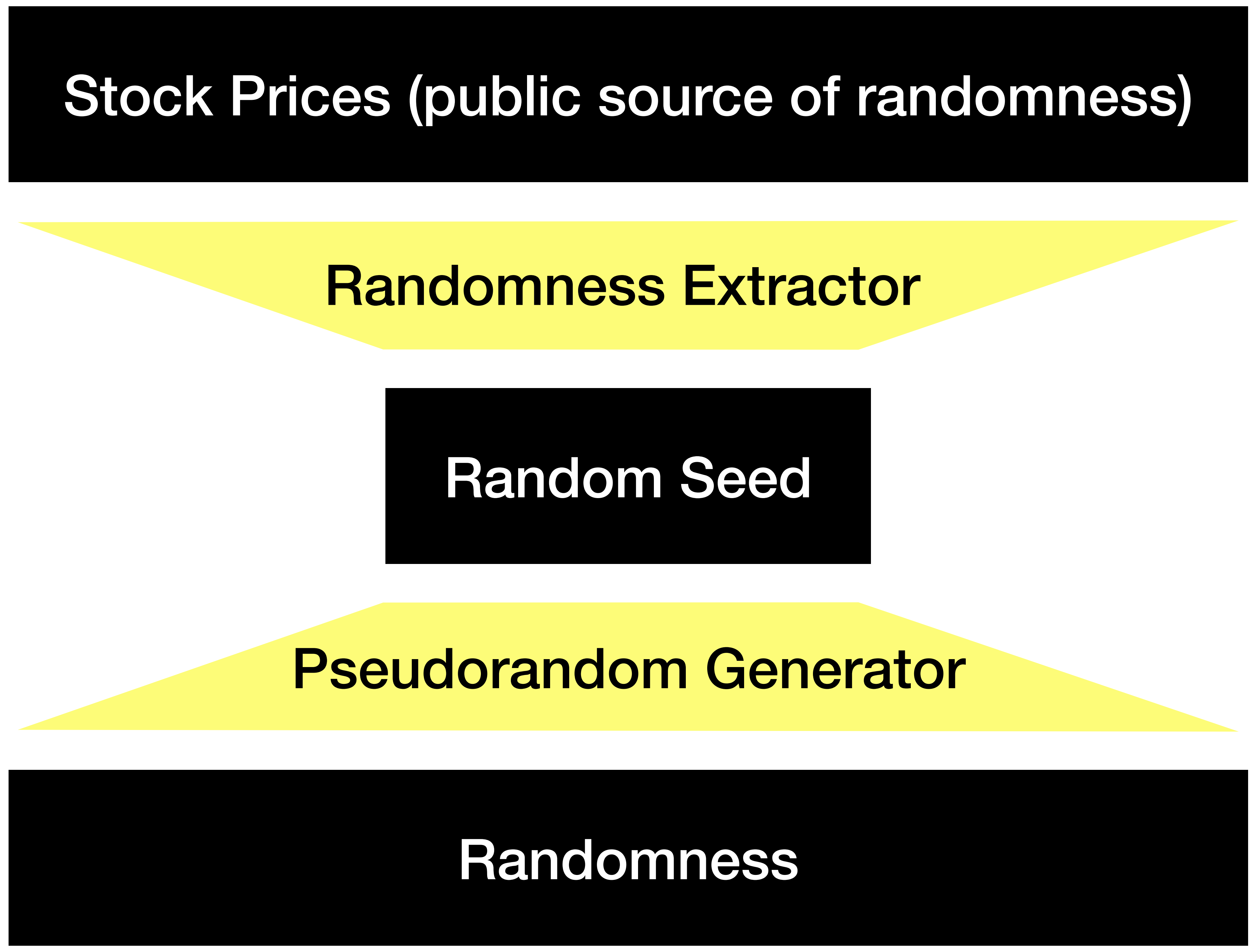
- Publish randomness regularly
- Cannot predict/manipulate



Why study VDF?

Randomness beacon

- Publish randomness regularly
- Cannot predict/manipulate

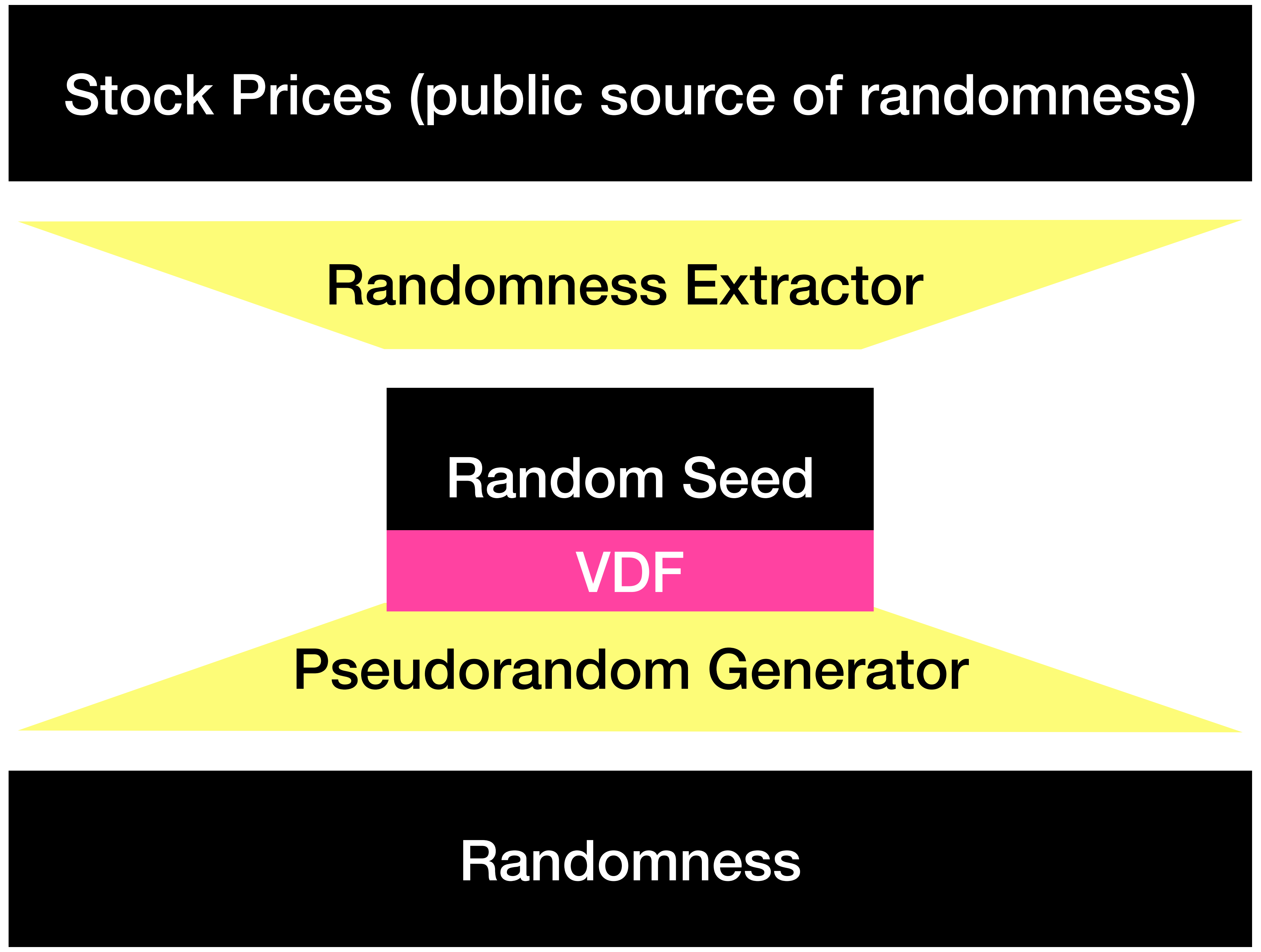


ISSUE: final randomness easy to compute & manipulate
(Stock prices can be biased/manipulated)

Why study VDF?

Randomness beacon

- Publish randomness regularly
- Cannot predict/manipulate

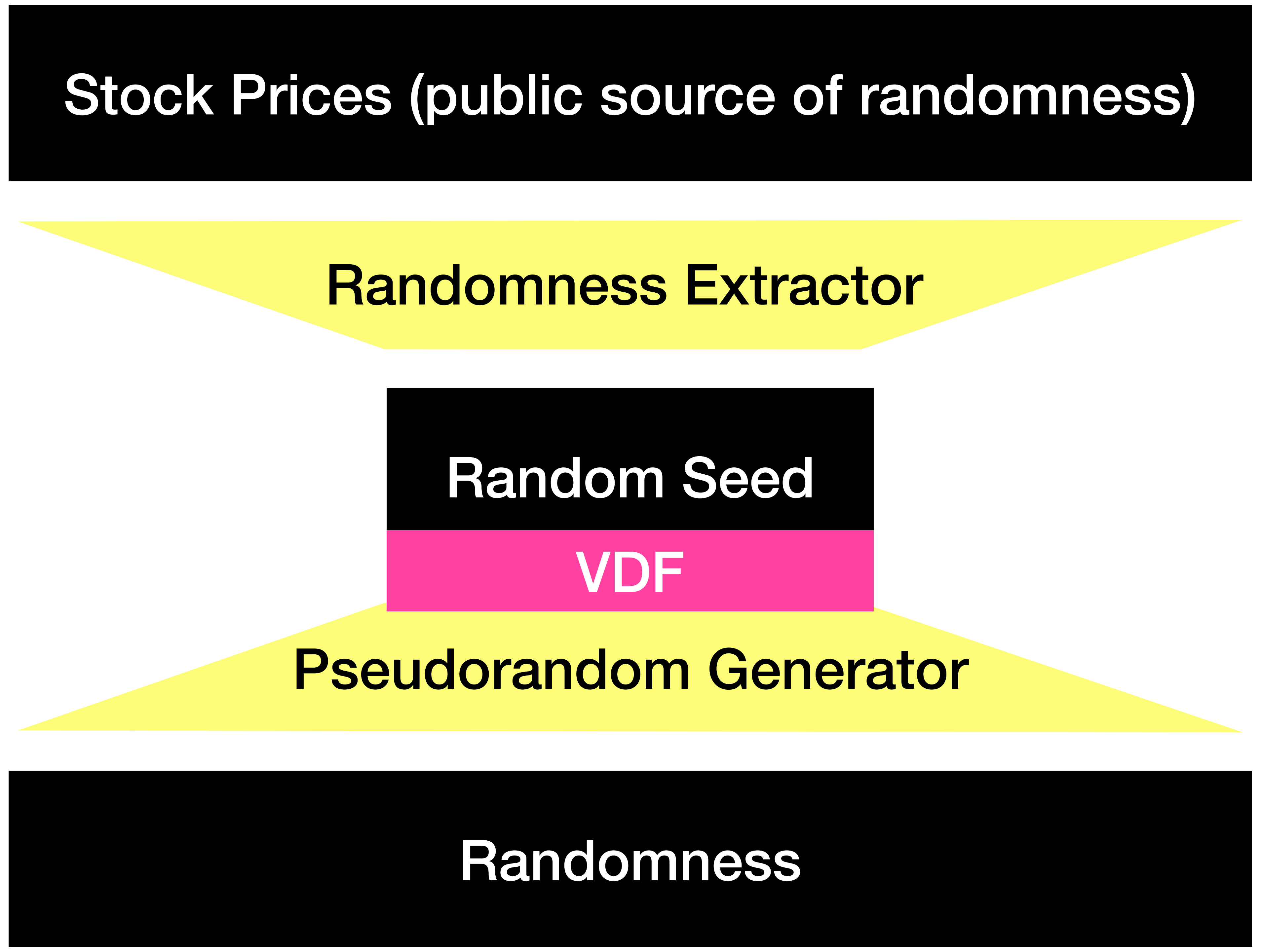


ISSUE: final randomness easy to compute & manipulate
(Stock prices can be biased/manipulated)

Why study VDF?

Randomness beacon

- Publish randomness regularly
- Cannot predict/manipulate



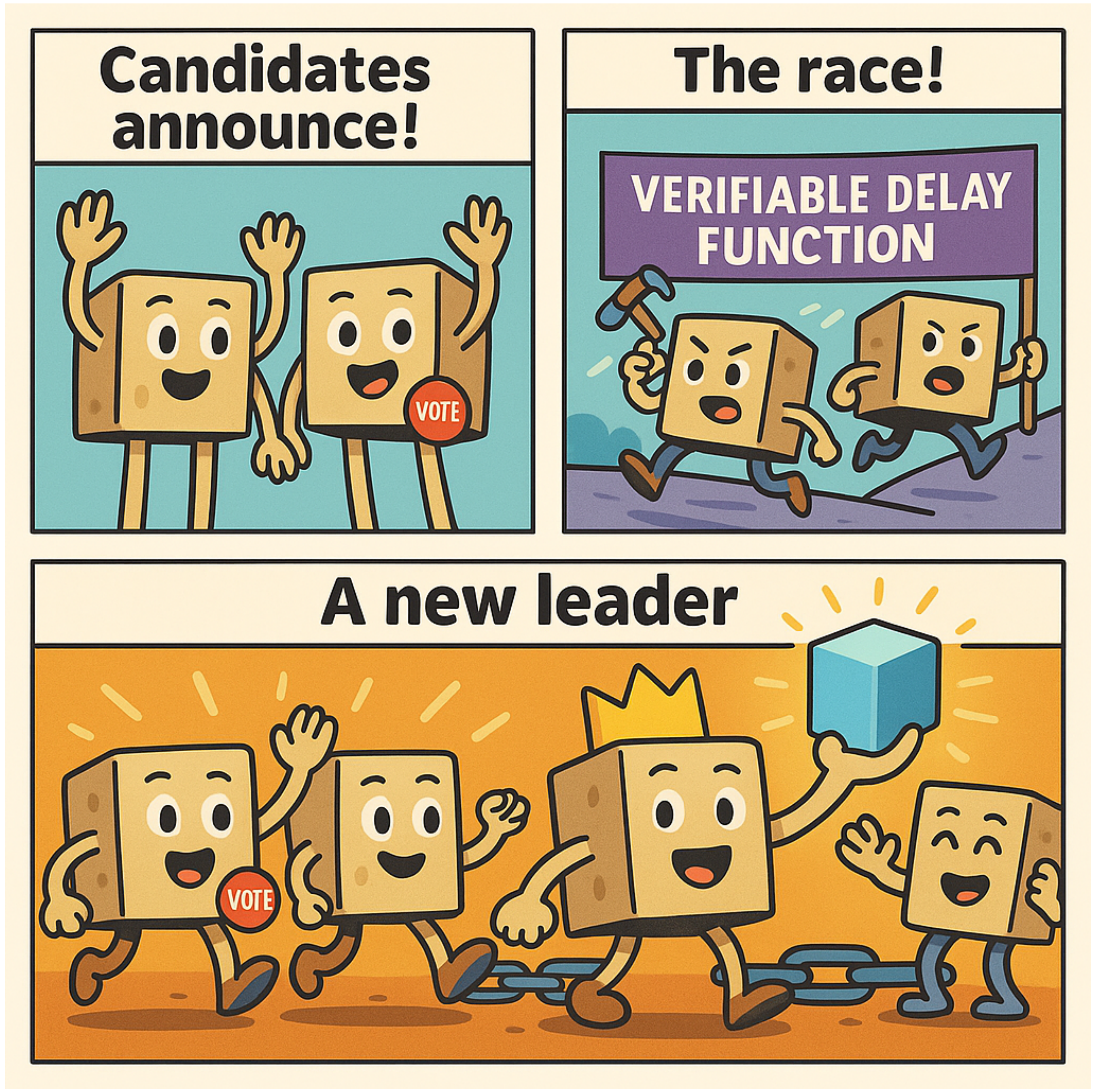
ISSUE: final randomness easy to compute & manipulate
(Stock prices can be biased/manipulated)

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…?

But, VDFs exist in the standard model…?

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

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?

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?

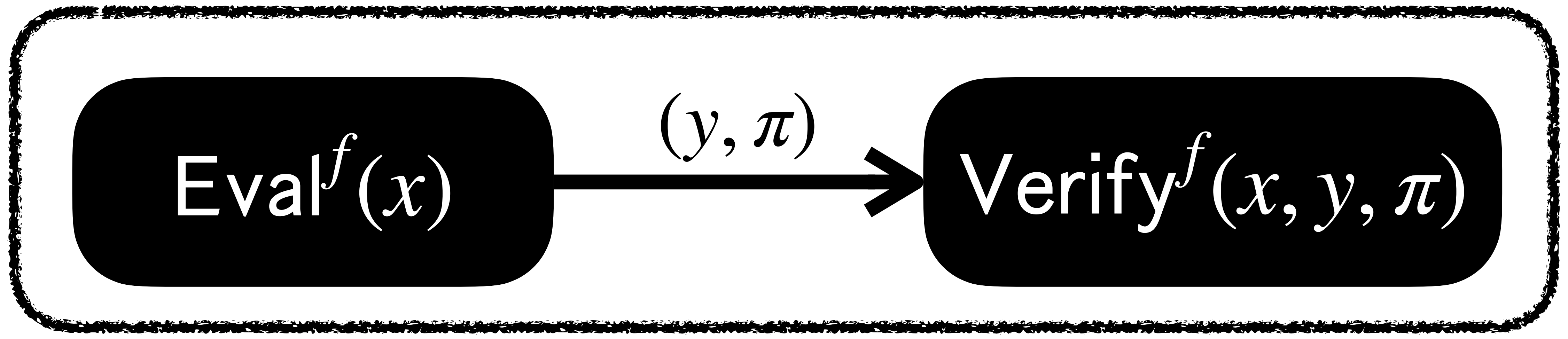
Existing VDFs rely on algebraic assumptions — not post-quantum secure

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



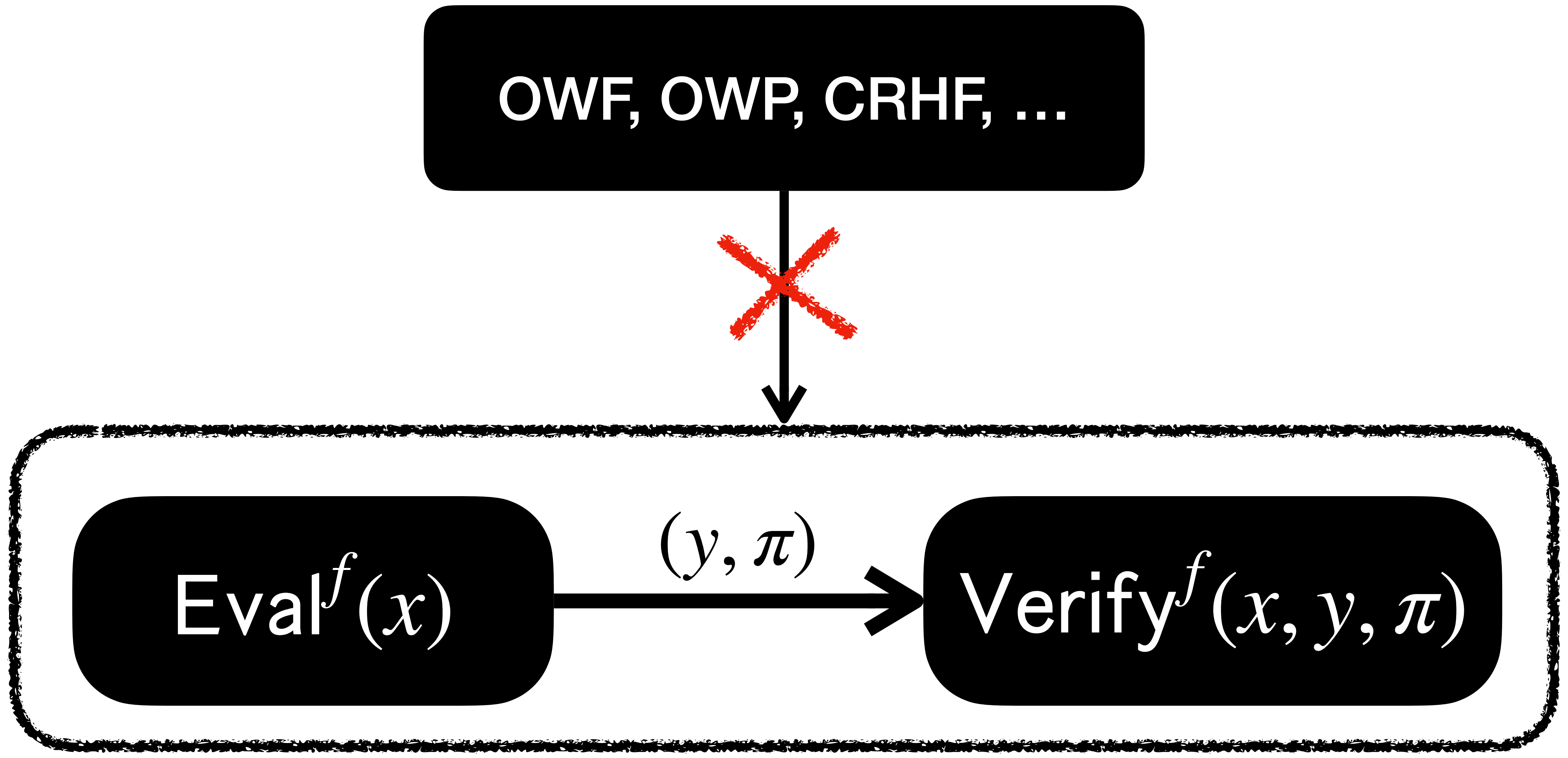
$$\begin{tikzcd}\boxed{\mathsf{Eval}^f(x)} \arrow[r, "(y,\pi)"] & \boxed{\mathsf{Verify}^f(x,y,\pi)}\end{tikzcd}$$

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

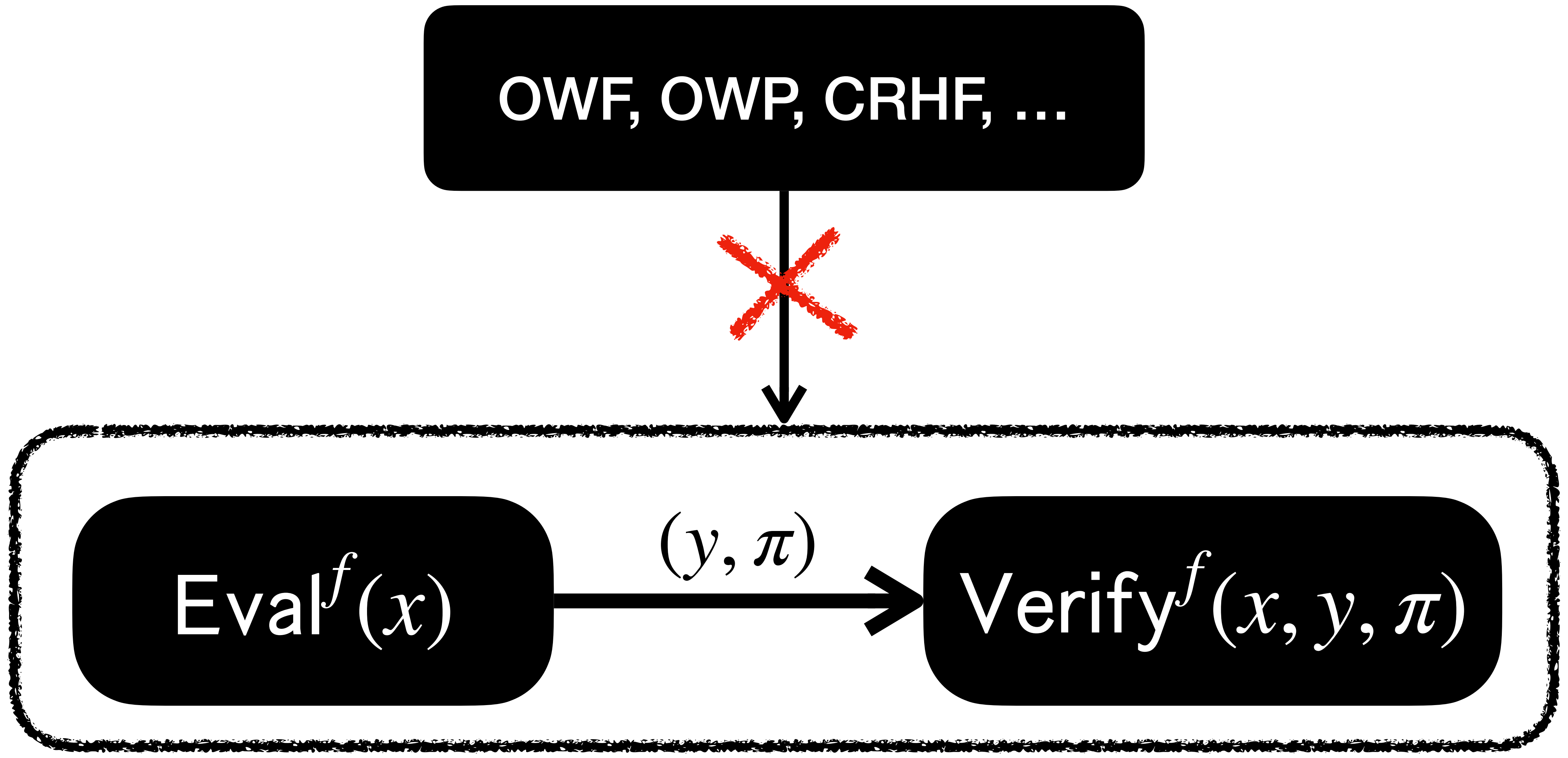


But, VDFs exist in the standard model…?

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

But, VDFs exist in the standard model…?

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

What security do VDF constructions have?

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…

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?

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)

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)

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

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

…

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

Query complexity vs. certificate complexity

Query complexity vs. certificate complexity

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

Query complexity vs. certificate complexity

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

e.g. $3\mathrm{WT}(a) = 1$ iff $\mathrm{wt}(a) \geq 3$

Query complexity vs. certificate complexity

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

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

e.g. $\mathsf{3WT}(a) = 1$ iff $\mathsf{wt}(a) \geq 3$

Query complexity vs. certificate complexity

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

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

e.g. $3\text{WT}(a) = 1$ iff $\text{wt}(a) \geq 3$
- $D(3\text{WT}) = n$

Query complexity vs. certificate complexity

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$
$C_1(G)$: min # bits in a needed to prove $G(a) = 1$
$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

e.g. $3\mathrm{WT}(a) = 1$ iff $\mathrm{wt}(a) \geq 3$
- $D(3\mathrm{WT}) = n$

Query complexity vs. certificate complexity

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$
$C_1(G)$: min # bits in a needed to prove $G(a) = 1$
$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

e.g. $3\text{WT}(a) = 1$ iff $\text{wt}(a) \geq 3$
- $D(3\text{WT}) = n$
- $C_1(3\text{WT}) = 3$
- $C_0(3\text{WT}) = n - 3$

Query complexity vs. certificate complexity

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$
$C_1(G)$: min # bits in a needed to prove $G(a) = 1$
$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

Trivial: $C(G) \leq D(G)$

e.g. 3WT$(a) = 1$ iff wt$(a) \geq 3$
- $D(\text{3WT}) = n$
- $C_1(\text{3WT}) = 3$
- $C_0(\text{3WT}) = n - 3$

Query complexity vs. certificate complexity

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$
$C_1(G)$: min # bits in a needed to prove $G(a) = 1$
$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

Trivial: $C(G) \le D(G)$

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

e.g. 3WT$(a) = 1$ iff wt$(a) \ge 3$
- $D(\text{3WT}) = n$
- $C_1(\text{3WT}) = 3$
- $C_0(\text{3WT}) = n - 3$

Verifiable ~~Delay~~ Function

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$
$C_1(G)$: min # bits in a needed to prove $G(a) = 1$
$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

Verifiable ~~Delay~~ Function

$$\begin{tikzcd}\boxed{\mathsf{Eval}^f(x)} \arrow[r, "y"] & \boxed{\mathsf{Verify}^f(x, y)}\end{tikzcd}$$

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$

$C_1(G)$: min # bits in a needed to prove $G(a) = 1$

$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

Verifiable ~~Delay~~ Function

$\mathsf{Eval}_x(f) := \mathsf{Eval}^f(x)$

$$\begin{tikzcd} \boxed{\mathsf{Eval}^f(x)} \arrow[r, "y"] & \boxed{\mathsf{Verify}^f(x, y)} \end{tikzcd}$$

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$

$C_1(G)$: min # bits in a needed to prove $G(a) = 1$

$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

Verifiable ~~Delay~~ Function

$$\begin{tikzcd} \boxed{\text{Eval}^f(x)} \arrow[r, "y"] & \boxed{\text{Verify}^f(x, y)} \end{tikzcd}$$

$\text{Eval}_x(f) := \text{Eval}^f(x)$

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

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$

$C_1(G)$: min # bits in a needed to prove $G(a) = 1$

$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

Verifiable ~~Delay~~ Function

$$\begin{tikzcd}\boxed{\mathsf{Eval}^f(x)} \arrow[r, "y"] & \boxed{\mathsf{Verify}^f(x, y)}\end{tikzcd}$$

$\mathsf{Eval}_x(f) := \mathsf{Eval}^f(x)$

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

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$

$C_1(G)$: min # bits in a needed to prove $G(a) = 1$

$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

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

Verifiable ~~Delay~~ Function

$$\begin{tikzcd}\boxed{\text{Eval}^f(x)} \arrow[r, "y"] & \boxed{\text{Verify}^f(x, y)}\end{tikzcd}$$

$\text{Eval}_x(f) := \text{Eval}^f(x)$

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

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$

$C_1(G)$: min # bits in a needed to prove $G(a) = 1$

$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

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

$\implies \exists$ algorithm with at most s^2 queries that computes Eval
i.e. breaking the "DELAY" requirement

Verifiable ~~Delay~~ Function

$$\begin{tikzcd} \text{Eval}^f(x) \arrow[r, "y"] & \text{Verify}^f(x, y) \end{tikzcd}$$

$\text{Eval}_x(f) := \text{Eval}^f(x)$

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

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

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

Certificate complexity $C(G) = \max\{C_1(G), C_0(G)\}$

$C_1(G)$: min # bits in a needed to prove $G(a) = 1$

$C_0(G)$: min # bits in a needed to prove $G(a) = 0$

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

$\Longrightarrow$ $\exists$ algorithm with at most s^2 queries that computes Eval
i.e. breaking the "DELAY" requirement

Challenge: VDF only has cryptographic correctness, above only works for statistical correctness…

Recap

Verifiable Delay Functions

Do Not Exist in the Random Oracle Model!!!!!!

Recap

Verifiable Delay Functions

Do Not Exist in the Random Oracle Model!!!!!!

Random Oracle Model	Delay	No delay
Unique	Impossible ❌	
Non-unique		

Recap

Verifiable Delay Functions

Do Not Exist in the Random Oracle Model!!!!!!

Random Oracle Model	Delay	No delay
Unique	Impossible ❌	
Non-unique		Proof of work ✅

Recap

Verifiable Delay Functions

Do Not Exist in the Random Oracle Model!!!!!!

Random Oracle Model	Delay	No delay
Unique	Impossible ❌	
Non-unique	Proof of sequential work ✅	Proof of work ✅

Recap

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

Random Oracle Model	Delay	No delay
Unique	Impossible ❌	???
Non-unique	Proof of sequential work ✅	Proof of work ✅

Recap

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

Random Oracle Model	Delay	No delay
Unique	Impossible ❌	???
Non-unique	Proof of sequential work ✅	Proof of work ✅

Thank you!