# **Tight** Verifiable Delay Functions

Nico Döttling Sanjam Garg Giulio Malavolta Prashant Vasudevan

# VERIFIABLE DELAY FUNCTIONS

A VDF [BBBF18] is a function f such that:

(Sequentiality) Requires time T to evaluate f(x)

(Verifiability) One can verify that f(x) = y in time << T

# VERIFIABLE DELAY FUNCTIONS

A VDF [BBBF18] is a function f such that:

(Sequentiality) Requires time T to evaluate f(x)

(Verifiability) One can verify that f(x) = y in time << T

Setup($1^\lambda$): Returns public parameters pp

Eval(pp,x,T): Returns output y and proof π

Verify(pp,x,y,π,T): Returns {0,1}

# EFFICIENCY OF VDFs

| | (parallel) Runtime |
|---|---|
| Setup($1^\lambda$): Returns public parameters pp | poly(λ) |
| Eval(pp,x,T): Returns output y and proof π | c·T |
| Verify(pp,x,y,π,T): Returns {0,1} | poly(λ,log(T)) |

# EFFICIENCY OF VDFs

| | (parallel) Runtime |
|---|---|
| Setup($1^\lambda$): Returns public parameters pp | poly(λ) |
| Eval(pp,x,T): Returns output y and proof π | c·T **tight** if c ≈ 1 |
| Verify(pp,x,y,π,T): Returns {0,1} | poly(λ,log(T)) |

# PROPERTIES OF VDFs

(Sequentiality) For all PRAM algorithms A with dept < T it holds that

$$(x, \mathrm{Eval}(pp, x, T)) \approx (x, \mathrm{uniform})$$

where $pp \leftarrow \mathrm{Setup}(1^{\lambda})$ and x is uniformly sampled.

# PROPERTIES OF VDFs

(Sequentiality) For all PRAM algorithms A with dept < T it holds that

$$(x,\mathrm{Eval}(pp,x,T)) \approx (x,\mathrm{uniform})$$

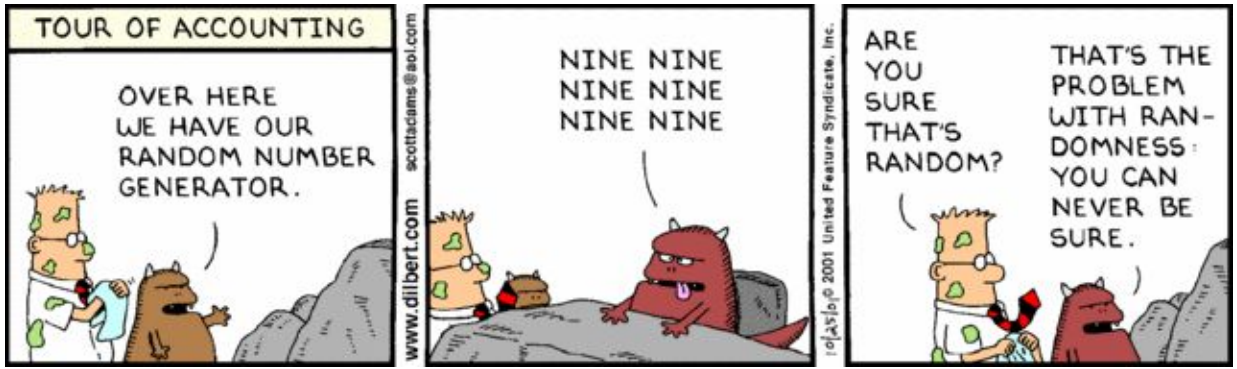
where $pp \leftarrow \mathrm{Setup}(1^{\lambda})$ and x is uniformly sampled.

(Soundness) For all PPT algorithms A it holds that

$$\Pr[\mathrm{Verify}(pp,x,y,\pi,T) = 1 \text{ AND } \mathrm{Eval}(pp,x,T) \neq y] \approx 0$$

where $(x,y,\pi,T) \leftarrow A(pp)$

# APPLICATION: COIN TOSSING



# COIN TOSSING (NO VDFs)

(Step 1) Each party locally samples some random coins $r_i$

(Step 2) Each party broadcasts $\mathrm{Com}(r_i)$

(Step 3) Each party outputs the opening of the commitment

(Step 4) Return $r = r_1 \oplus \ldots \oplus r_n$

# COIN TOSSING (NO VDFs)

(Step 1) Each party locally samples some random coins $r_i$

(Step 2) Each party broadcasts $\mathrm{Com}(r_i)$

(Step 3) Each party outputs the opening of the commitment

(Step 4) Return $r = r_1 \oplus \ldots \oplus r_n$

**Unbiased?**

# COIN TOSSING (NO VDFs)

(Step 1) Each party locally samples some random coins $r_i$

(Step 2) Each party broadcasts $\mathrm{Com}(r_i)$

(Step 3) Each party outputs the opening of the commitment

(Step 4) Return $r = r_1 \oplus \ldots \oplus r_n$

**Unbiased? NO: The last party to open can abort to bias r**

# COIN TOSSING (WITH VDFs)

(Step 1) Each party locally samples some random coins $r_i$

(Step 2) Each party broadcasts $r_i$ *in plain*

(Step 3) Return $r = \mathrm{Eval}(pp, r_1 \,||\, \ldots \,||\, r_n, T)$

where T = time-out parameter

# COIN TOSSING (WITH VDFs)

(Step 1) Each party locally samples some random coins $r_i$

(Step 2) Each party broadcasts $r_i$ *in plain*

(Step 3) Return $r = \text{Eval}(pp, r_1 \,||\, \ldots \,||\, r_n, T)$

where T = time-out parameter

**Unbiased?**

# COIN TOSSING (WITH VDFs)

(Step 1) Each party locally samples some random coins $r_i$

(Step 2) Each party broadcasts $r_i$ *in plain*

(Step 3) Return $r = \text{Eval}(pp, r_1 \,||\, \ldots \,||\, r_n, T)$

where T = time-out parameter

**Unbiased? YES: By the sequentiality of VDF**

# STATE-OF-THE-ART

| | ASSUMPTIONS | PROOF SIZE* | PROVER COMPLEXITY* |
|---|---|---|---|
| SNARGs | Random Oracle | O(1) | $c \cdot T$ ($c > 1$) |
| Incremental VC | Random Oracle (non BlackBox) | O(1) | Tight! |
| Pietrzak | Sequential Squaring | log(T) | $\sqrt{T}$ |
| Wesolowski | Sequential Squaring + Adaptive Root | O(1) | Tight! |

*Suppressing factors in $\lambda$

# STATE-OF-THE-ART

| | ASSUMPTIONS | PROOF SIZE* | PROVER COMPLEXITY |
|---|---|---|---|
| SNARGs | Random Oracle | O(1) | $c \cdot T$ ($c > 1$) |
| Incremental VC | Random Oracle (non BlackBox) | O(1) | Tight! |
| Pietrzak | Sequential Squaring | log(T) | $\sqrt{T}$ |
| Wesolowski | Sequential Squaring + Adaptive Root | O(1) | Tight! |

# OUR RESULTS

(Theorem 1) If there exists a VDF with prover runtime of $c \cdot T$ (for any constant c), then there exists a *tight* VDF.

(Theorem 2) There exists no *blackbox* construction of a VDF in the random oracle model where the prover makes at most $T + O(T^d)$ queries (for $d < 1$).

Concurrent work by Mahmoody, Smith, and Wu [ICALP 2020]

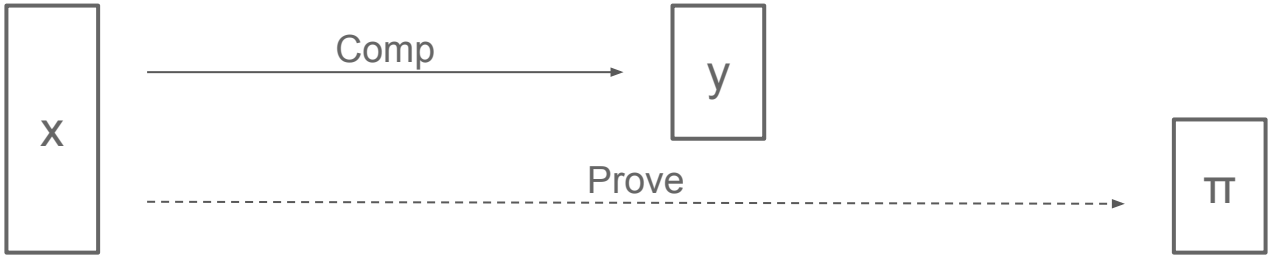
# OUR RESULTS

(Theorem 1) If there exists a VDF with prover runtime of $c \cdot T$ (for any constant c), then there exists a *tight* VDF.

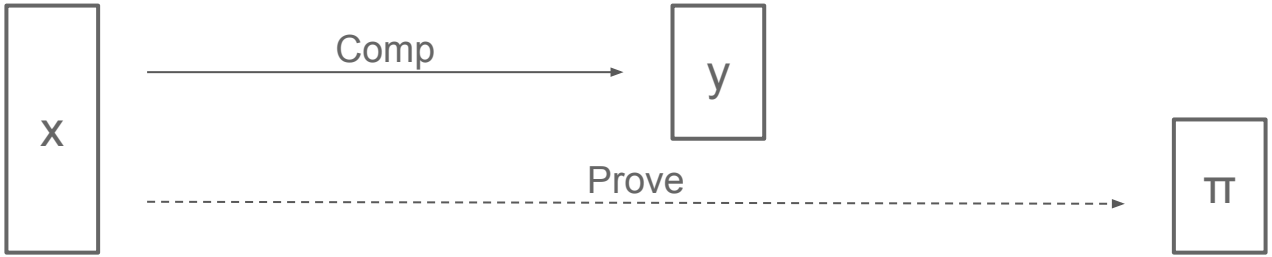
(Theorem 2) There exists no *blackbox* construction of a VDF in the random oracle model where the prover makes at most $T + O(T^d)$ queries (for $d < 1$).

Concurrent work by Mahmoody, Smith, and Wu [ICALP 2020]

# SELF-COMPOSABLE VDFs



# SELF-COMPOSABLE VDFs



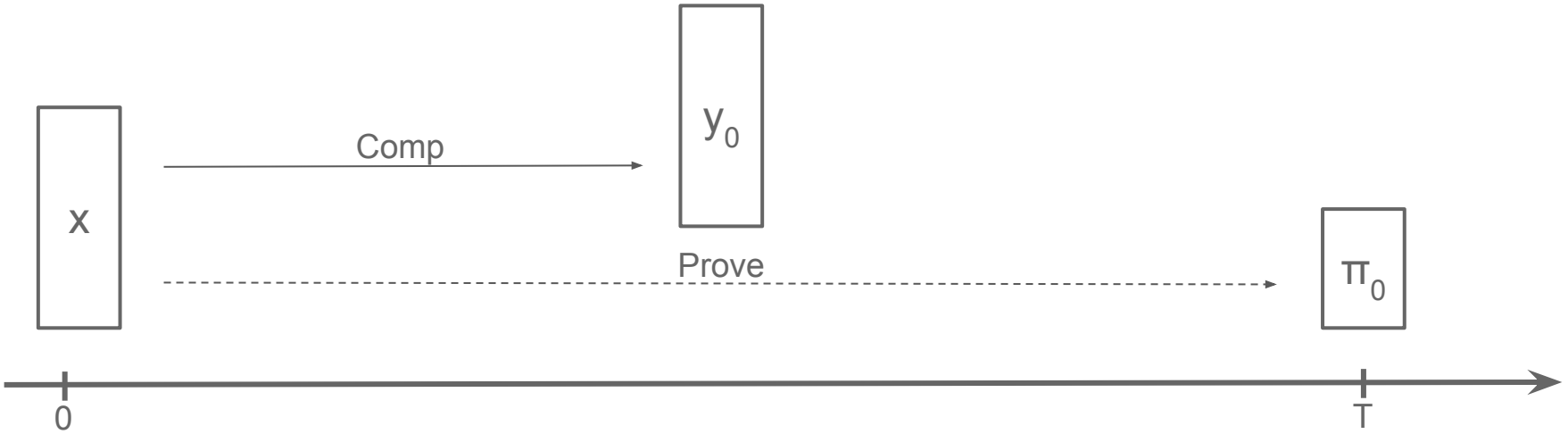

(Self-composability) It holds that

$$\text{Comp}(pp, x, T_0+T_1) = \text{Comp}(pp, \text{Comp}(pp, x, T_0), T_1)$$

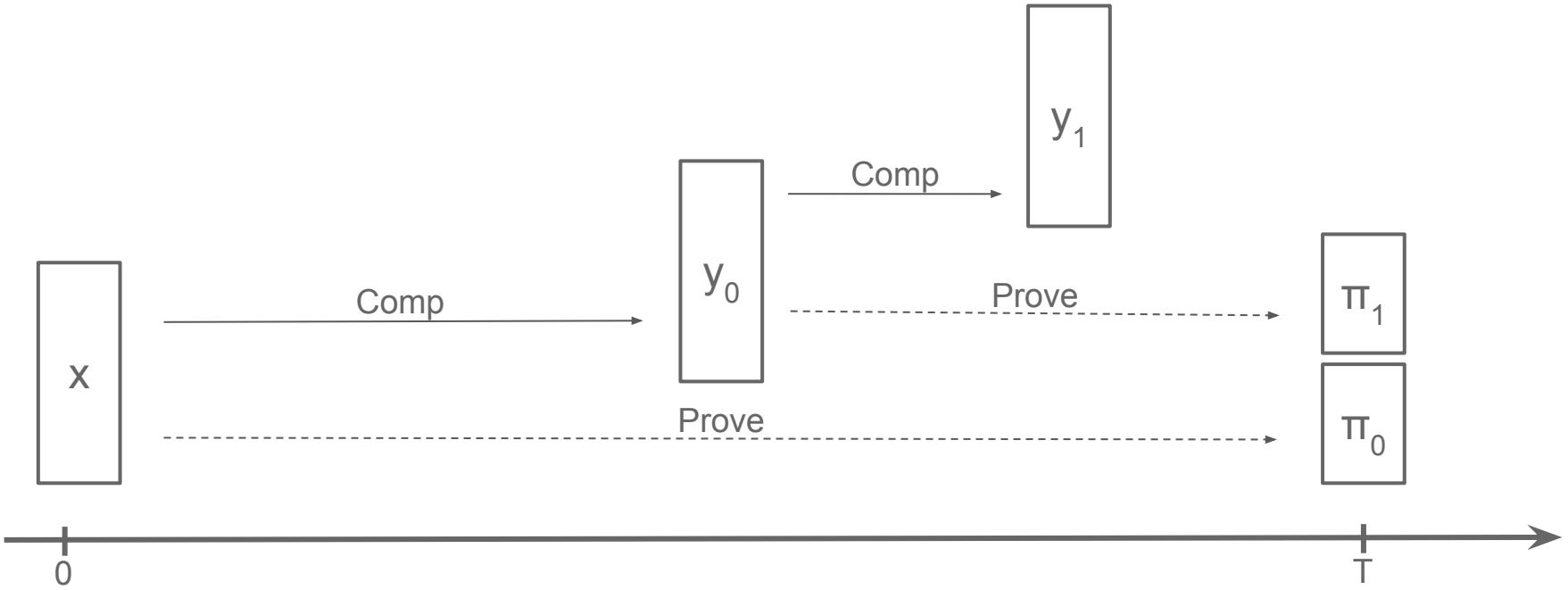
# TIGHT VDFs (simp.)

$|\text{Comp}| = T$ and $|\text{Prove}| = 2 \cdot T$

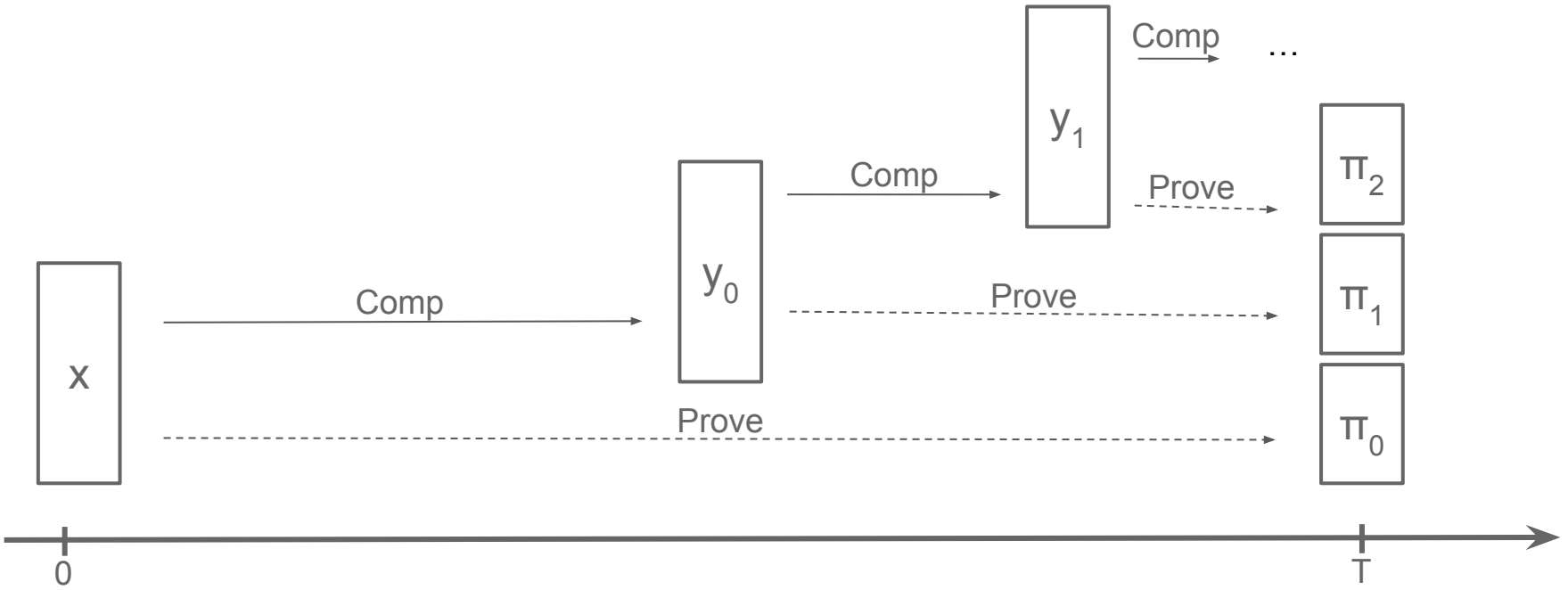
# TIGHT VDFs (simp.)



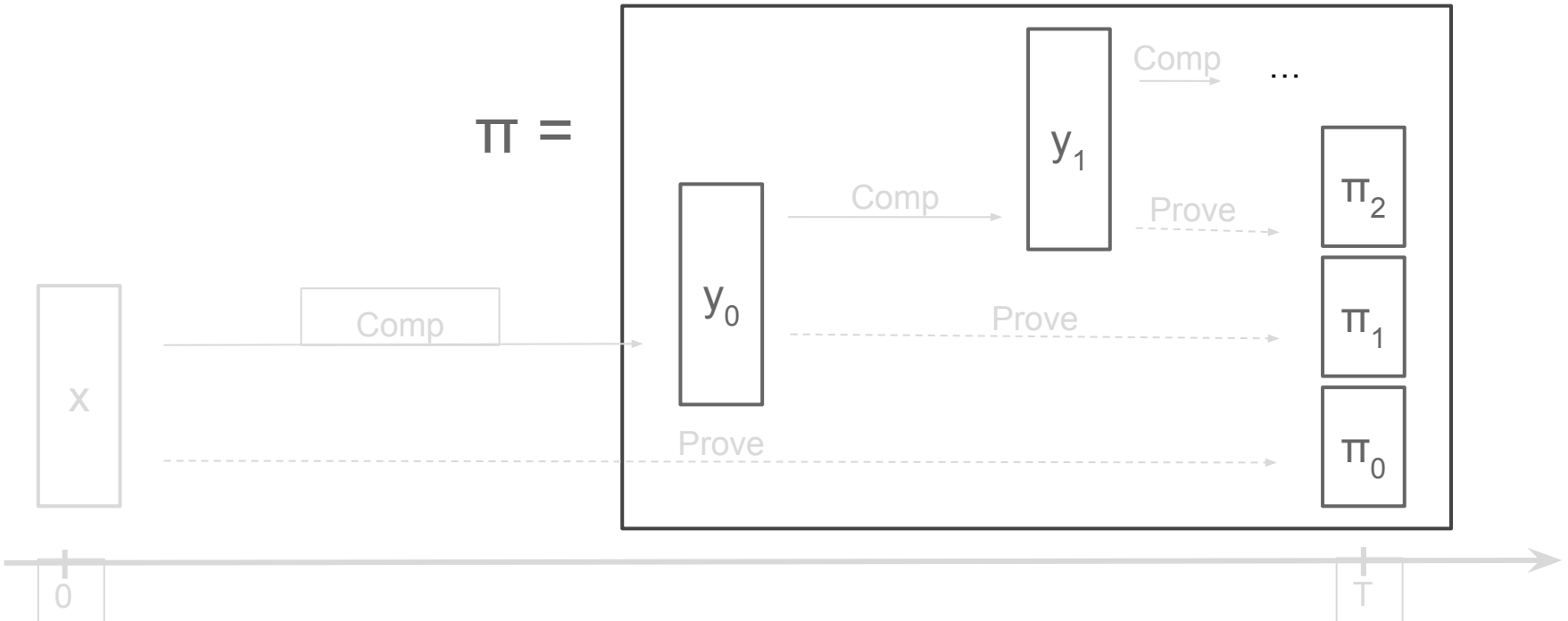
# TIGHT VDFs (simp.)



# TIGHT VDFs (simp.)



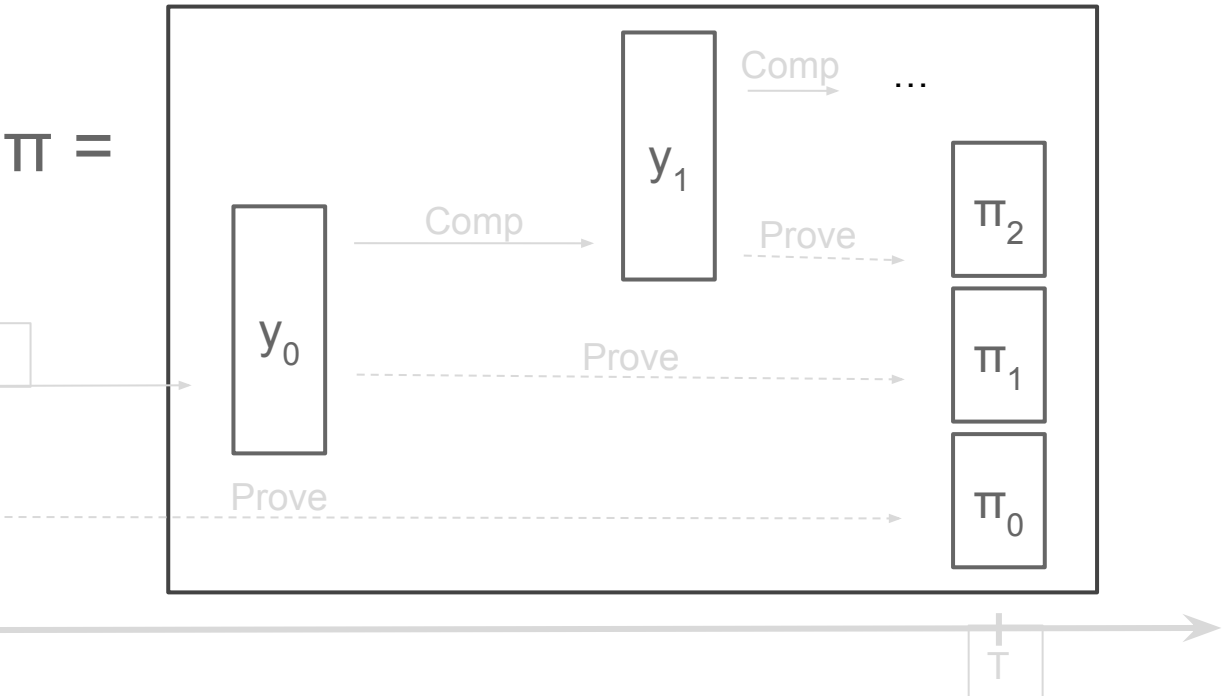
# TIGHT VDFs (simp.)



# TIGHT VDFs (simp.)

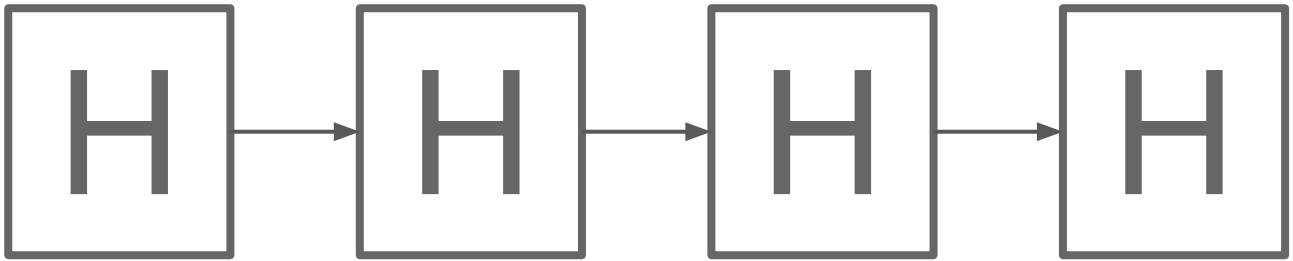
Prover runtime ≈ T

Increase of ~log(T) in:
- Proof Size
- Prover Parallelism



# NEW IMPLICATIONS

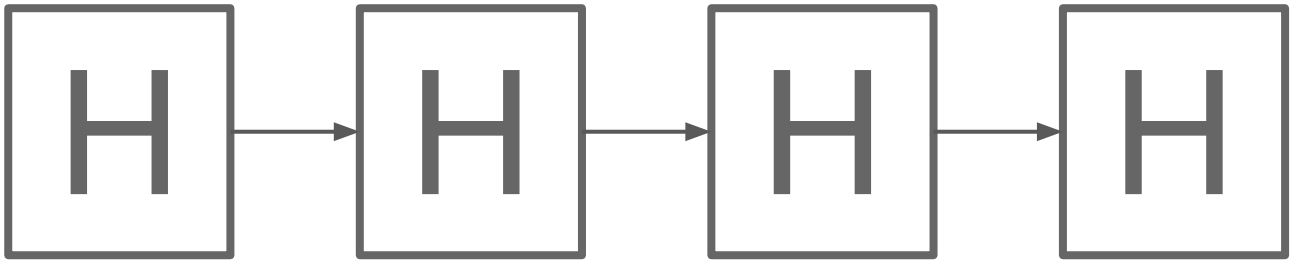
(1) Tight VDFs from repeated hashing + SNARGs




+ SNARG

# NEW IMPLICATIONS

(1) Tight VDFs from repeated hashing + SNARGs




+ SNARG

(2) Tight Pietrzak's VDFs

=> Tight VDFs from sequential squaring (and statistical soundness in RO)

# CONCLUSIONS AND OPEN PROBLEMS

In this work:

- A bootstrapping theorem for **tight** VDFs
- A lower bound for **blackbox** constructions

# CONCLUSIONS AND OPEN PROBLEMS

In this work:

- A bootstrapping theorem for **tight** VDFs
- A lower bound for **blackbox** constructions

Open problems:

- Efficient post-quantum VDFs?
- Efficient VDF in the standard model?