Separating Verifiable Delay Functions and Time-Lock Puzzles

Max von Consbruch* and Nivesh Aggarwal†

Joint work with Hamza Abusalah‡, Karen Azari* and Chethan Kamath†

*Uni Vienna
†IIT Bombay
‡IMDEA Madrid

Main Result

Theorem 1 (Main Result)

1. *Verifiable Delay Functions do not fully black-box imply Time-Lock Puzzles.*
2. *Time-Lock Puzzles do not fully black-box imply Verifiable Delay Functions.*

Main Result

Theorem 1 (Main Result)

1. *Verifiable Delay Functions do not fully black-box imply Time-Lock Puzzles.*
2. *Time-Lock Puzzles do not fully black-box imply Verifiable Delay Functions.*

This Talk:

1. Definition of VDFs and TLPs

Main Result

Theorem 1 (Main Result)

1. *Verifiable Delay Functions do not fully black-box imply Time-Lock Puzzles.*
2. *Time-Lock Puzzles do not fully black-box imply Verifiable Delay Functions.*

This Talk:

1. Definition of VDFs and TLPs
2. Short excourse on black-box reductions

Main Result

Theorem 1 (Main Result)

1. *Verifiable Delay Functions do not fully black-box imply Time-Lock Puzzles.*
2. *Time-Lock Puzzles do not fully black-box imply Verifiable Delay Functions.*

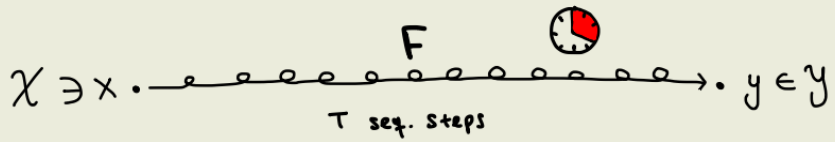
This Talk:

1. Definition of VDFs and TLPs
2. Short excourse on black-box reductions
3. Proof strategy

Verifiable Delay Functions (VDFs)

A *Verifiable Delay Function* is

- a delay function $F\colon \mathcal{X} \to \mathcal{Y}$

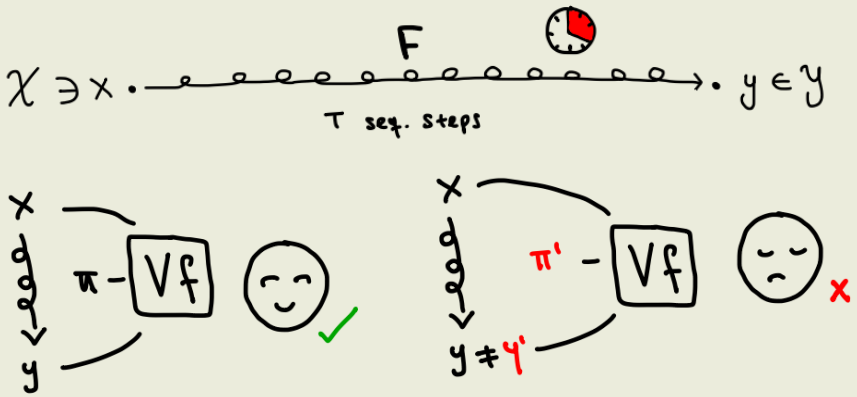


Sequentiality: for random $x \leftarrow X$, no adversary with depth $\ll T$ can compute $F(x)$

Verifiable Delay Functions (VDFs)

A *Verifiable Delay Function* is

- a delay function $F : \mathcal{X} \to \mathcal{Y}$
- ... which is verifiable!



Sequentiality: for random $x \leftarrow X$, no adversary with depth $\ll T$ can compute $F(x)$

Verifiable Delay Functions (VDFs)

A *Verifiable Delay Function* is

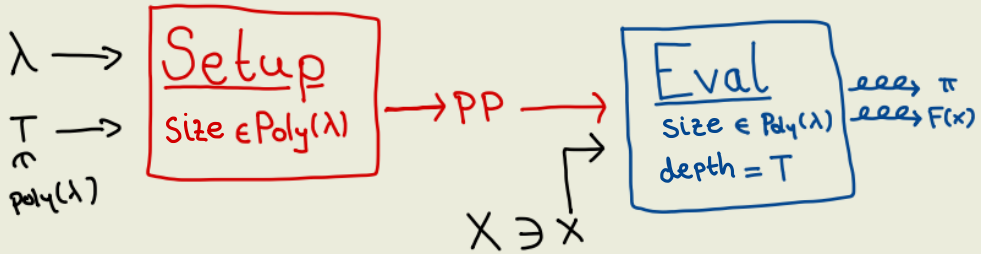
- a delay function $F\colon \mathcal{X} \to \mathcal{Y}$
- ... which is verifiable!
- More precisely, it is a triple of algorithms $\Pi_{\mathrm{VDF}} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ with the following syntax

Sequentiality: for random $x \leftarrow X$, no adversary with depth $\ll T$ can compute $F(x)$

Verifiable Delay Functions (VDFs)

A *Verifiable Delay Function* is

- a delay function $F\colon \mathcal{X} \to \mathcal{Y}$
- ... which is verifiable!
- More precisely, it is a triple of algorithms $\Pi_{\mathrm{VDF}} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ with the following syntax

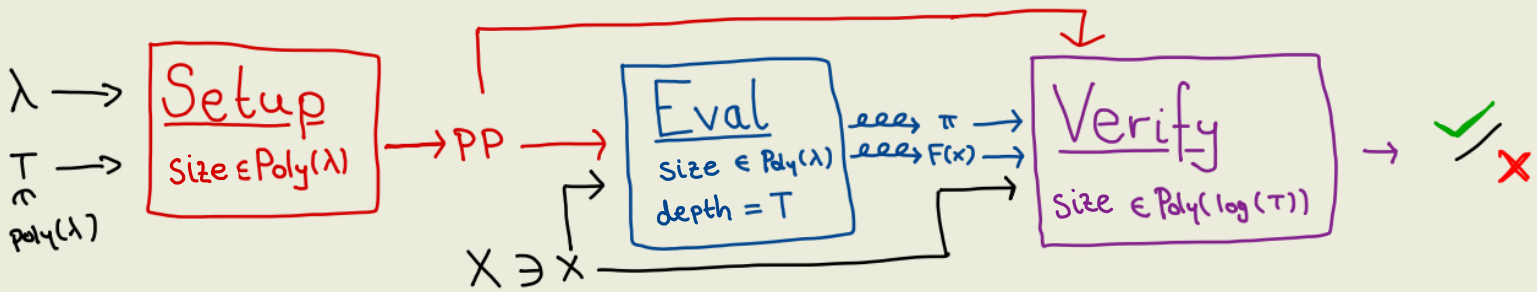


Sequentiality: for random $x \leftarrow X$, no adversary with depth $\ll T$ can compute $F(x)$

Verifiable Delay Functions (VDFs)

A *Verifiable Delay Function* is

- a delay function $F : \mathcal{X} \to \mathcal{Y}$
- ... which is verifiable!
- More precisely, it is a triple of algorithms $\Pi_{\mathrm{VDF}} = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ with the following syntax



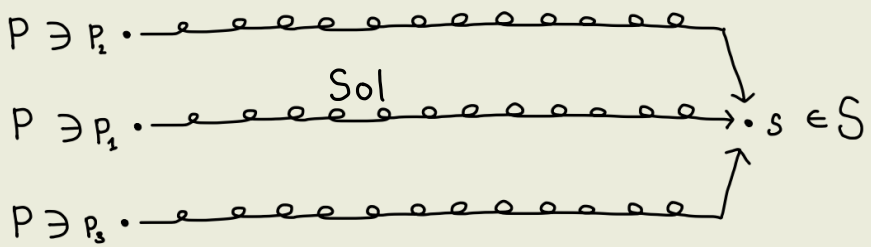
Sequentiality: for random $x \leftarrow X$, no adversary with depth $\ll T$ can compute $F(x)$

Soundness: given (pp, x), no PPT adversary can find (y', π') s.t. $y' \neq F(x)$ and $\mathsf{Verify}(\mathsf{pp}, x, y', \pi') = 1$.

Time-Lock Puzzles (TLPs)

A *Time-Lock Puzzle* is a pair of algorithms $\Pi_{\mathsf{TLP}} = (\mathsf{PGen}, \mathsf{Sol})$ having

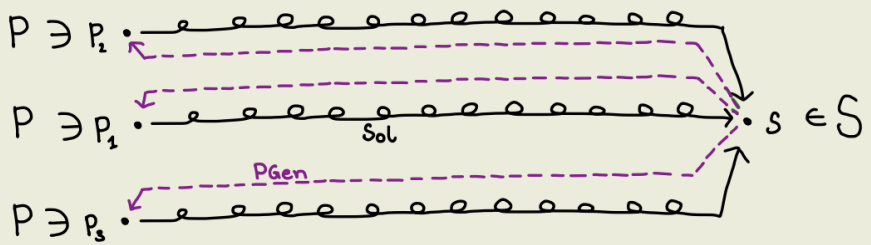
- a many-to one delay function $\mathsf{Sol}\colon \mathcal{P} \to \mathcal{S}$



Time-Lock Puzzles (TLPs)

A *Time-Lock Puzzle* is a pair of algorithms $\Pi_{\mathsf{TLP}} = (\mathsf{PGen}, \mathsf{Sol})$ having

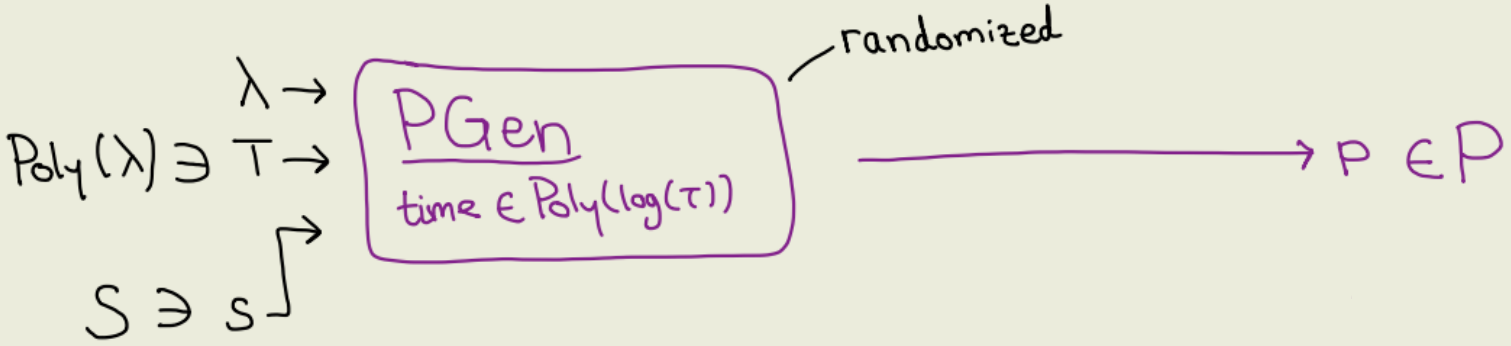
- a many-to one delay function $\mathsf{Sol}\colon \mathcal{P} \to \mathcal{S}$
- ... which allows for quick (randomized) generation of preimages



Time-Lock Puzzles (TLPs)

A *Time-Lock Puzzle* is a pair of algorithms $\Pi_{\mathsf{TLP}} = (\mathsf{PGen}, \mathsf{Sol})$ having

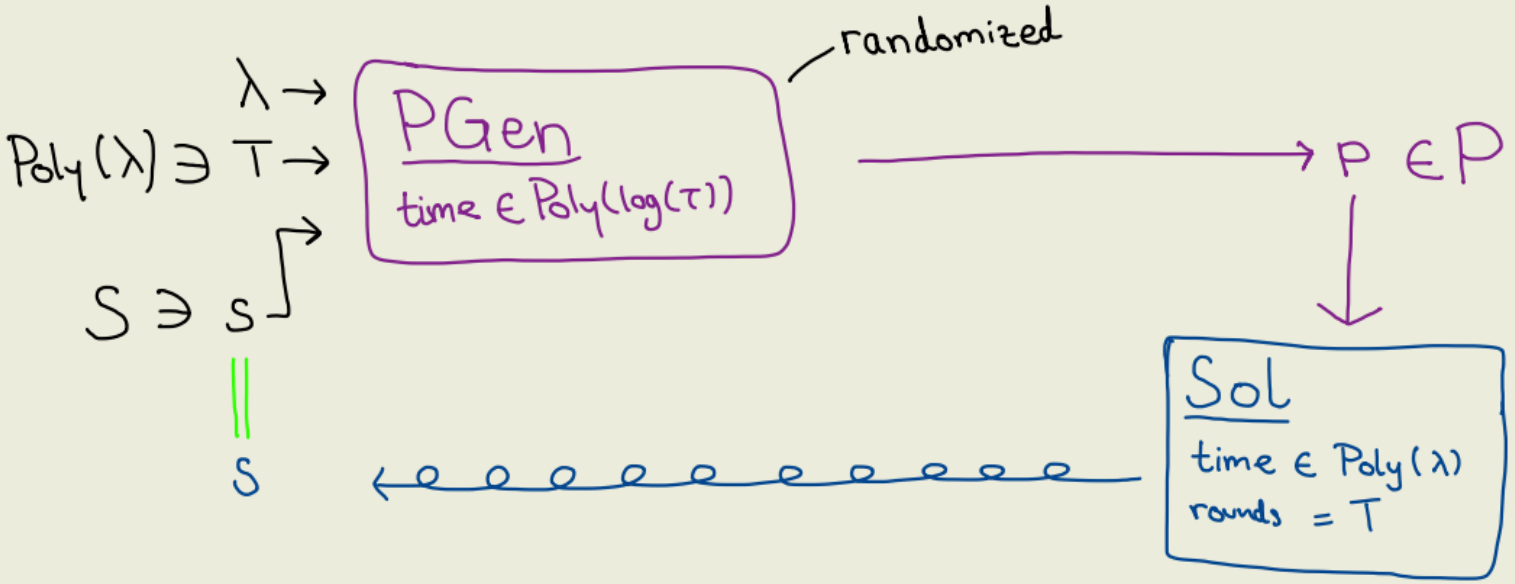
- a many-to one delay function $\mathsf{Sol}\colon \mathcal{P} \to \mathcal{S}$
- ... which allows for quick (randomized) generation of preimages
- More precisely, it is pair of algorithms $\Pi_{\mathsf{TLP}} = (\mathsf{PGen}, \mathsf{Sol})$ with the following syntax



Time-Lock Puzzles (TLPs)

A *Time-Lock Puzzle* is a pair of algorithms $\Pi_{\mathsf{TLP}} = (\mathsf{PGen}, \mathsf{Sol})$ having

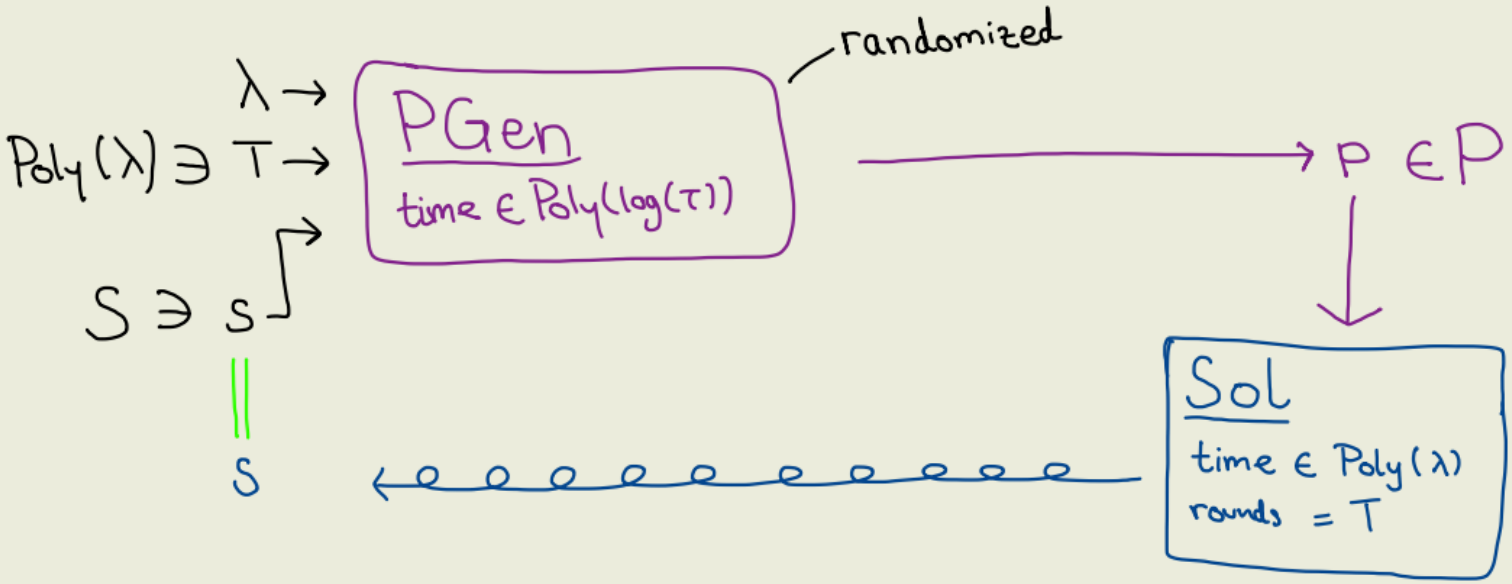
- a many-to one delay function $\mathsf{Sol}\colon \mathcal{P} \to \mathcal{S}$
- ... which allows for quick (randomized) generation of preimages
- More precisely, it is pair of algorithms $\Pi_{\mathsf{TLP}} = (\mathsf{PGen}, \mathsf{Sol})$ with the following syntax



Time-Lock Puzzles (TLPs)

A *Time-Lock Puzzle* is a pair of algorithms $\Pi_{\mathsf{TLP}} = (\mathsf{PGen}, \mathsf{Sol})$ having

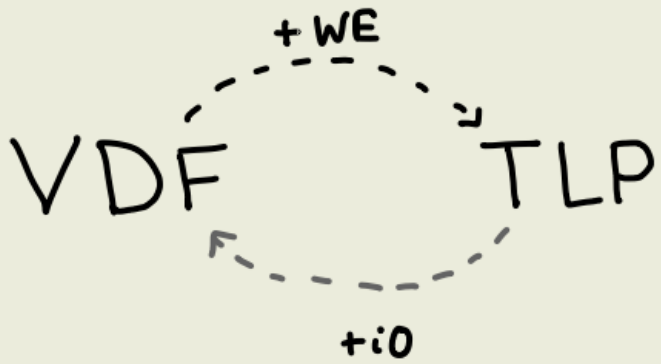
- a many-to one delay function $\mathsf{Sol}\colon \mathcal{P} \to \mathcal{S}$
- ... which allows for quick (randomized) generation of preimages
- More precisely, it is pair of algorithms $\Pi_{\mathsf{TLP}} = (\mathsf{PGen}, \mathsf{Sol})$ with the following syntax



Indistinguishability of Puzzles. For any $s_0, s_1 \in \mathcal{S}$,
$\mathsf{PGen}(s_0, T) \approx_{\mathrm{depth} \ll T} \mathsf{PGen}(s_1, T)$

Known Implications

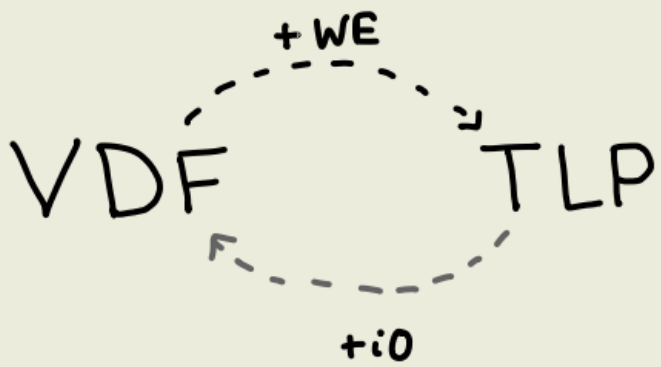
Known Implications VDF $\leftrightarrow$ TLP:



Known Implications

Known Implications VDF $\leftrightarrow$ TLP:

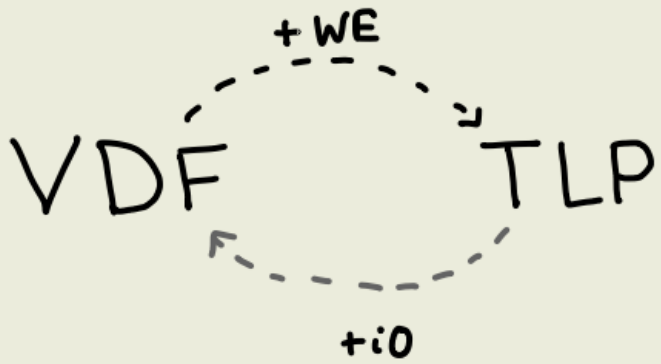
- [Ren20]: VDFs + (extractable) witness encryption $\rightarrow$ TLPs



Known Implications

Known Implications VDF $\leftrightarrow$ TLP:

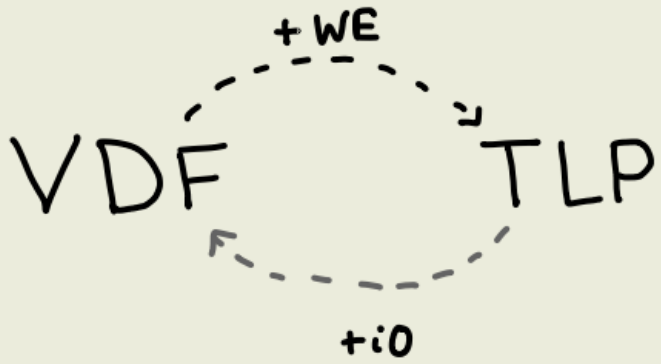
- [Ren20]: VDFs + (extractable) witness encryption $\rightarrow$ TLPs
- [AAFKT25]: TLPs + indistinguishability obfuscation $\rightarrow$ "weak" VDFs



Known Implications

Known Implications VDF $\leftrightarrow$ TLP:

- [Ren20]: VDFs + (extractable) witness encryption $\rightarrow$ TLPs
- [AAFKT25]: TLPs + indistinguishability obfuscation $\rightarrow$ "weak" VDFs



However, these constructions

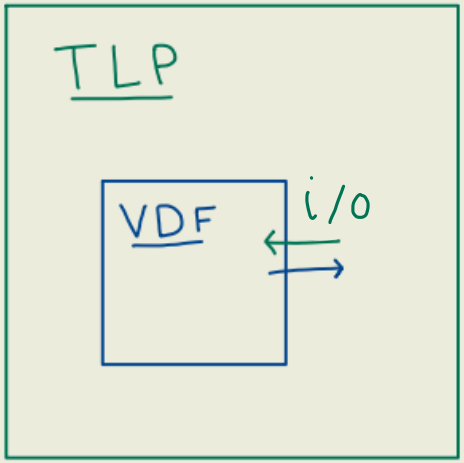
- rely on additional assumptions
- are *not* black-box

Fully Black-Box Reductions

Example. A fully black-box construction VDF $\rightarrow$ TLP would be...

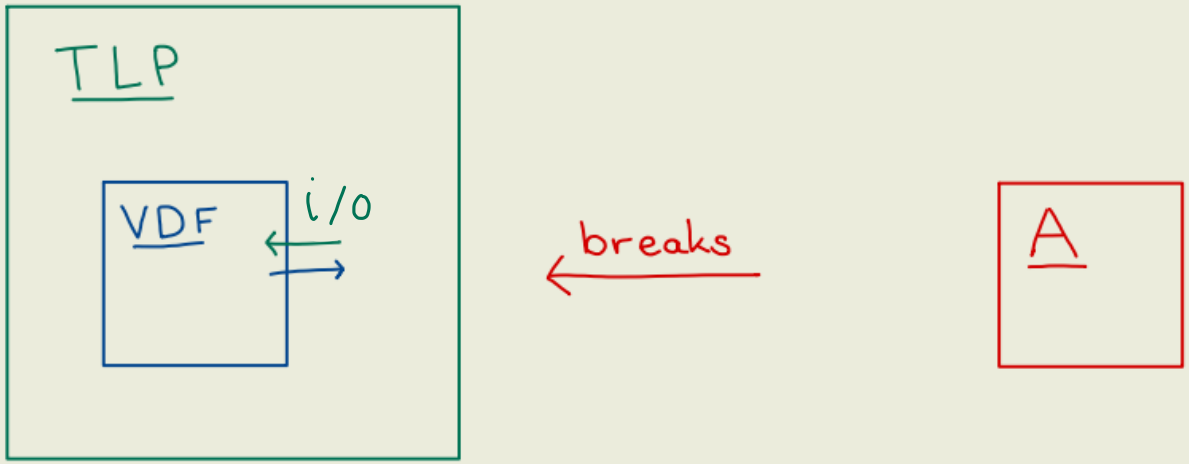
Fully Black-Box Reductions

Example. A fully black-box construction VDF $\rightarrow$ TLP would be...



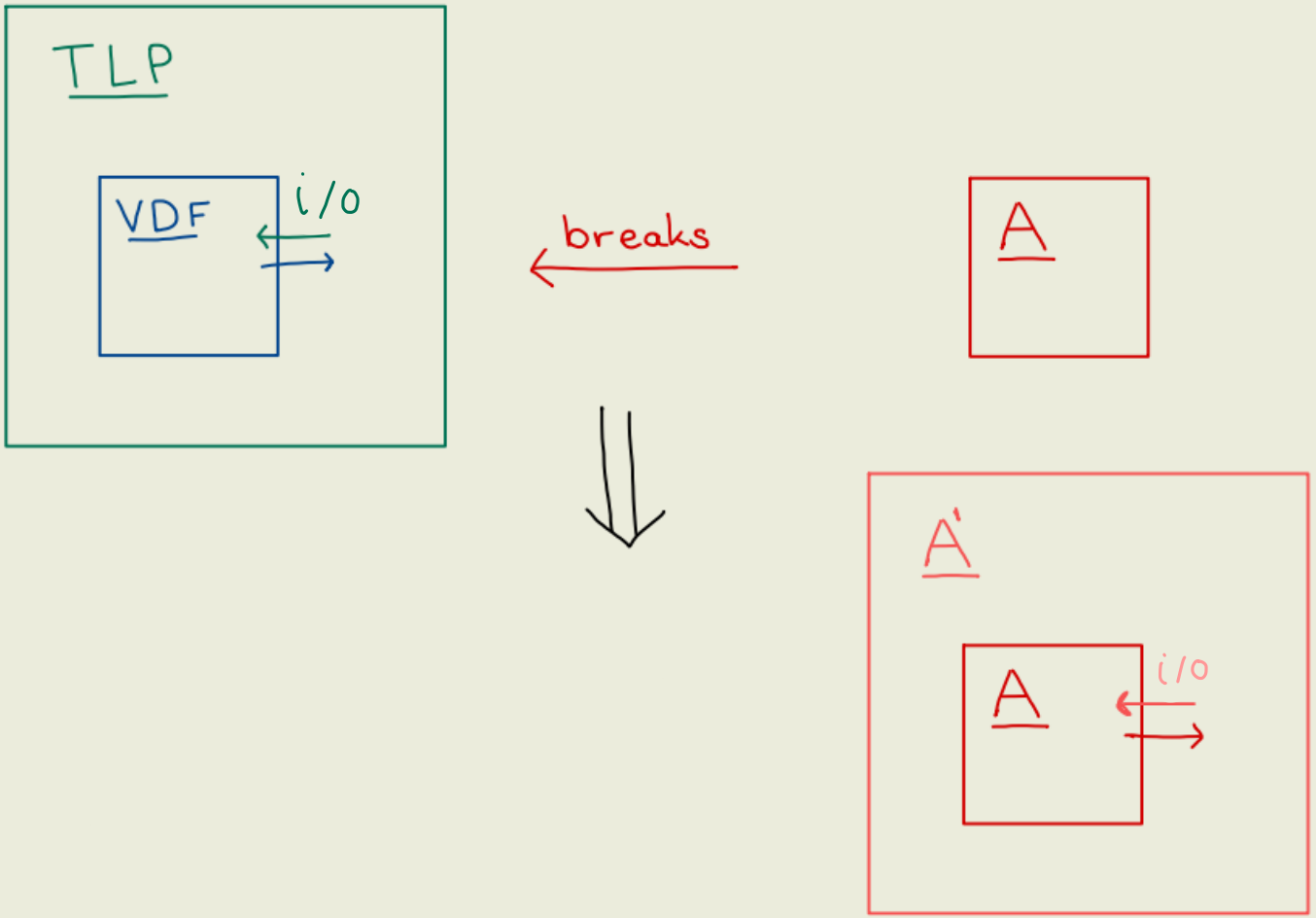
Fully Black-Box Reductions

Example. A fully black-box construction VDF $\to$ TLP would be...



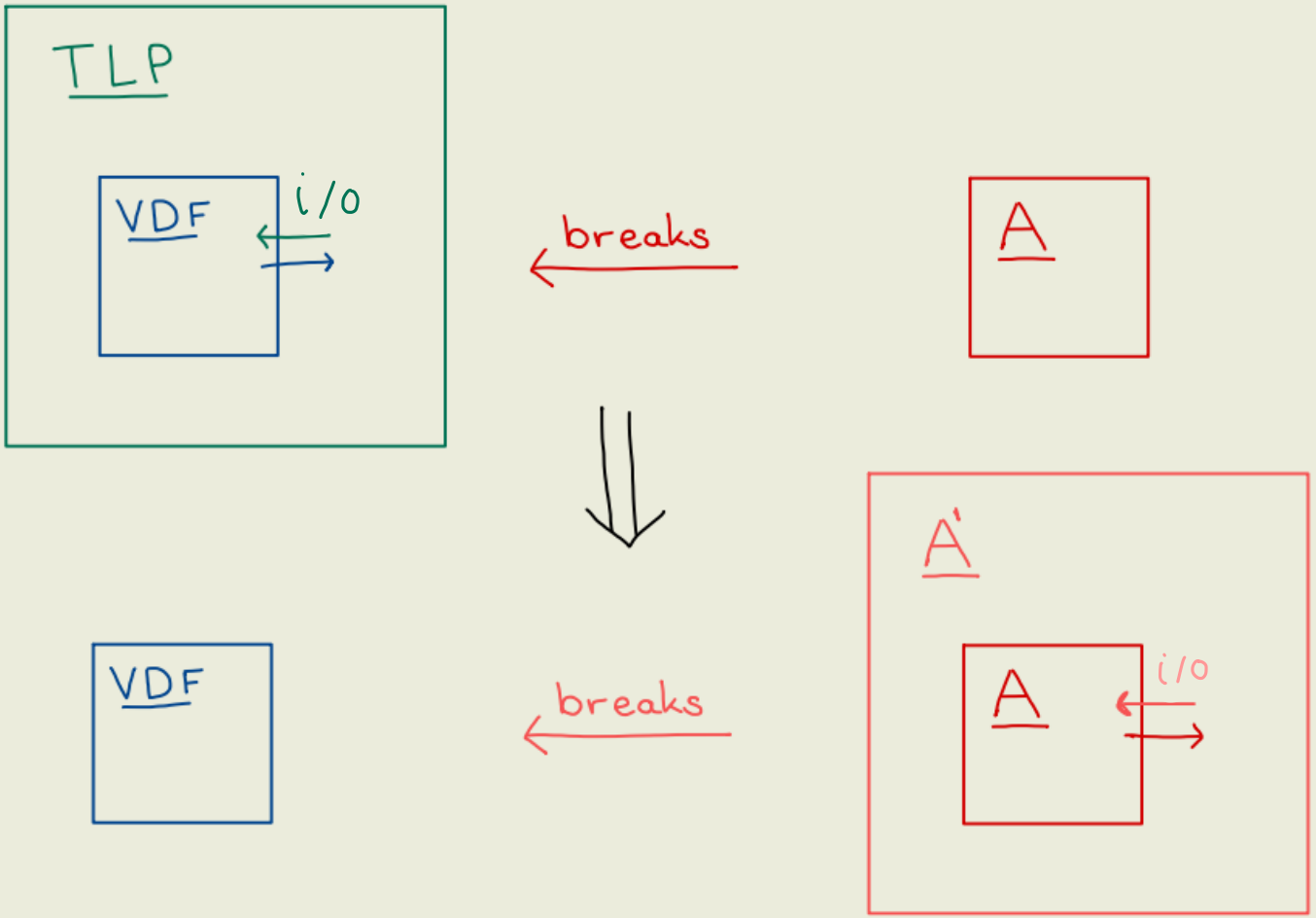
Fully Black-Box Reductions

Example. A fully black-box construction VDF $\to$ TLP would be...



Fully Black-Box Reductions

Example. A fully black-box construction VDF $\to$ TLP would be...



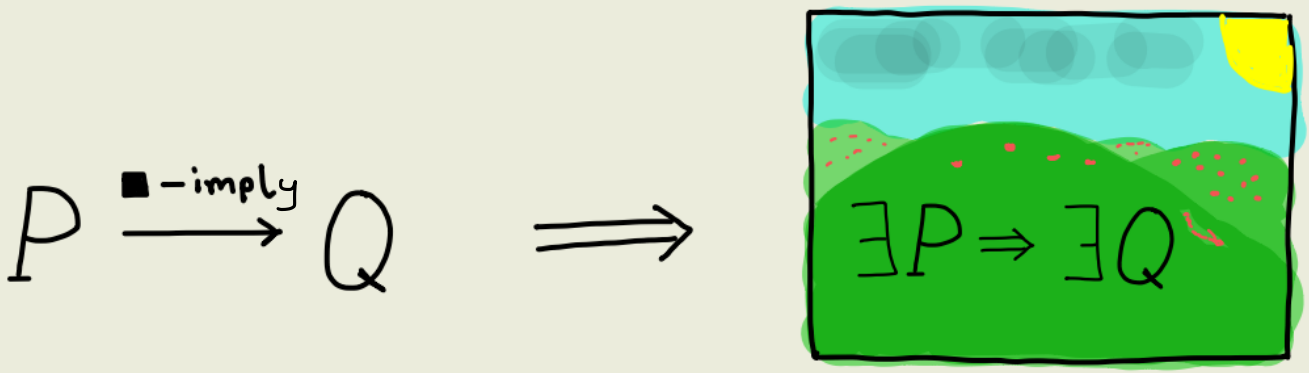
Ruling out Black-Box Reductions

Main Point: Fully Black-Box Constructions relativize

Ruling out Black-Box Reductions

Main Point: Fully Black-Box Constructions relativize

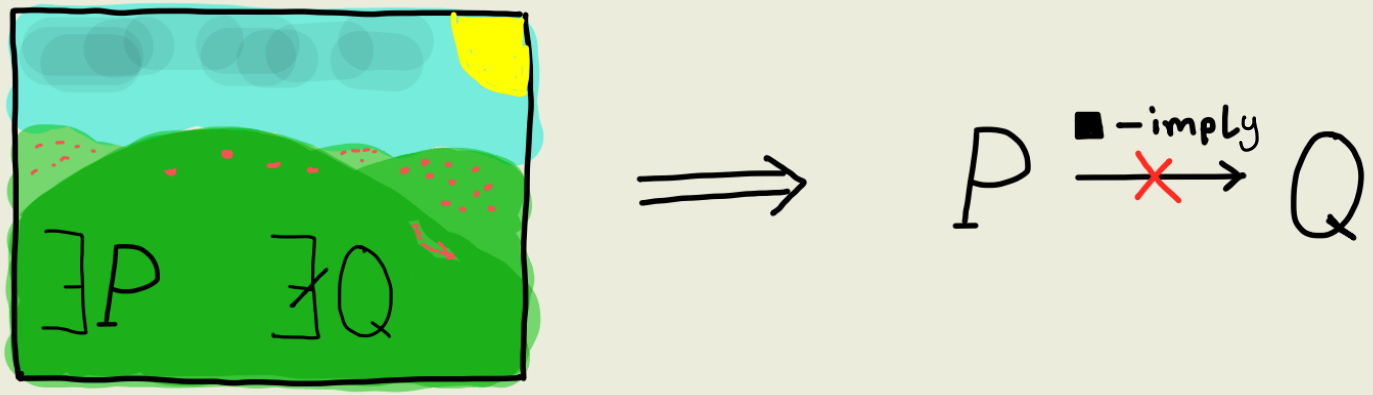
If P black-box implies Q, then in every "world" in which P exists, Q exists.



Ruling out Black-Box Reductions

Main Point: Fully Black-Box Constructions relativize

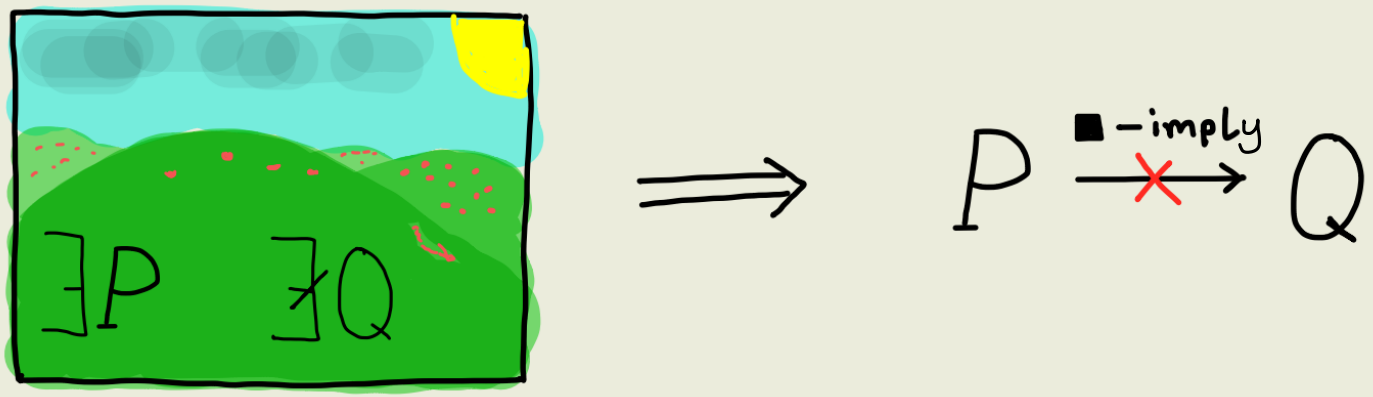
If there is a "world" in which P exists and Q does not exist, P does not black-box imply Q



Ruling out Black-Box Reductions

Main Point: Fully Black-Box Constructions relativize

If there is a "world" in which P exists and Q does not exist, P does not black-box imply Q



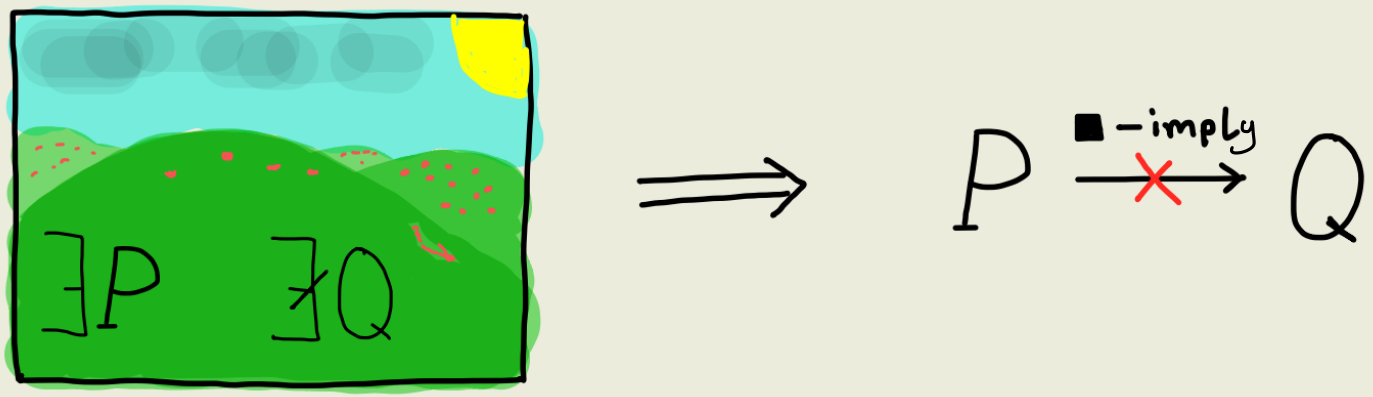
We rule out black-box constructions TLP $\rightarrow$ VDF and VDF $\rightarrow$ TLP by constructing "worlds"

- $\mathcal{M}_{\mathrm{VDF}}$, where VDFs exist but TLPs do not.

Ruling out Black-Box Reductions

Main Point: Fully Black-Box Constructions relativize

If there is a "world" in which P exists and Q does not exist, P does not black-box imply Q



We rule out black-box constructions TLP $\rightarrow$ VDF and VDF $\rightarrow$ TLP by constructing "worlds"

- $\mathcal{M}_{\mathrm{VDF}}$, where VDFs exist but TLPs do not.
- $\mathcal{M}_{\mathrm{TLP}}$, where TLPs exist but VDFs do not.

Constructing $\mathcal{M}_{\mathrm{VDF}}$ & $\mathcal{M}_{\mathrm{TLP}}$

Starting point: (parallel) Random Oracle Model (ROM)

Constructing $\mathcal{M}_{\mathrm{VDF}}$ & $\mathcal{M}_{\mathrm{TLP}}$

Starting point: (parallel) Random Oracle Model (ROM)

- All algorithms have access to a random function f

Constructing $\mathcal{M}_{\mathrm{VDF}}$ & $\mathcal{M}_{\mathrm{TLP}}$

Starting point: (parallel) Random Oracle Model (ROM)

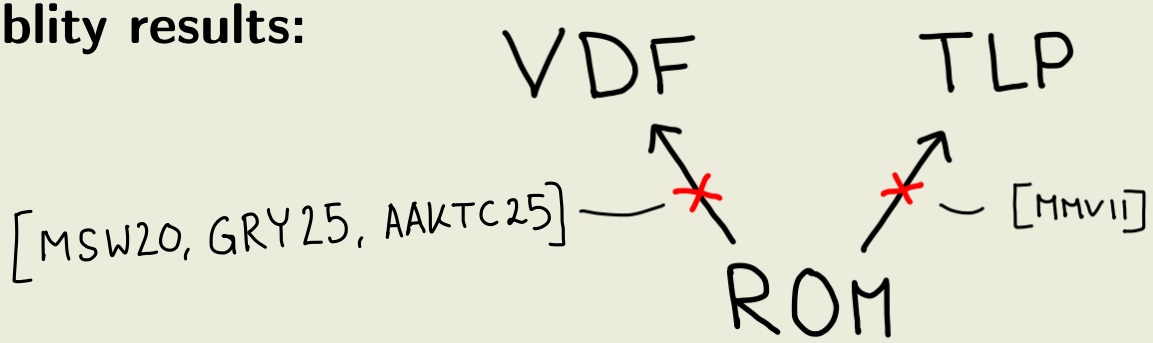
- All algorithms have access to a random function f
- Complexity is measured exclusively by oracle queries
  - total #queries
  - #rounds of queries

Constructing $\mathcal{M}_{\mathrm{VDF}}$ & $\mathcal{M}_{\mathrm{TLP}}$

Starting point: (parallel) Random Oracle Model (ROM)

- All algorithms have access to a random function f
- Complexity is measured exclusively by oracle queries

We have impossiblity results:

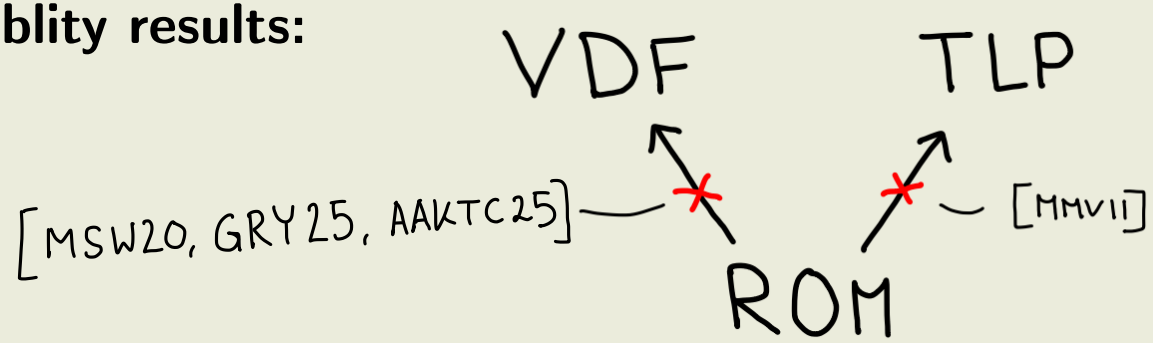


Constructing $\mathcal{M}_{\mathrm{VDF}}$ & $\mathcal{M}_{\mathrm{TLP}}$

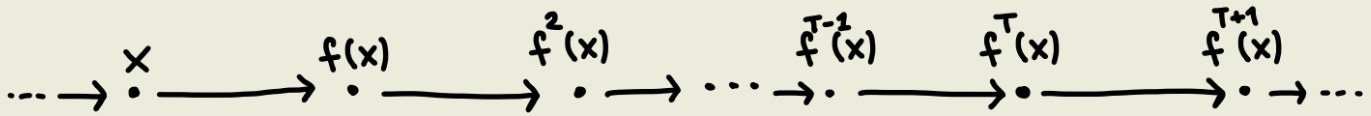
Starting point: (parallel) Random Oracle Model (ROM)

- All algorithms have access to a random function f
- Complexity is measured exclusively by oracle queries

We have impossiblity results:



But: there is a simple delay function $F = f^T$:

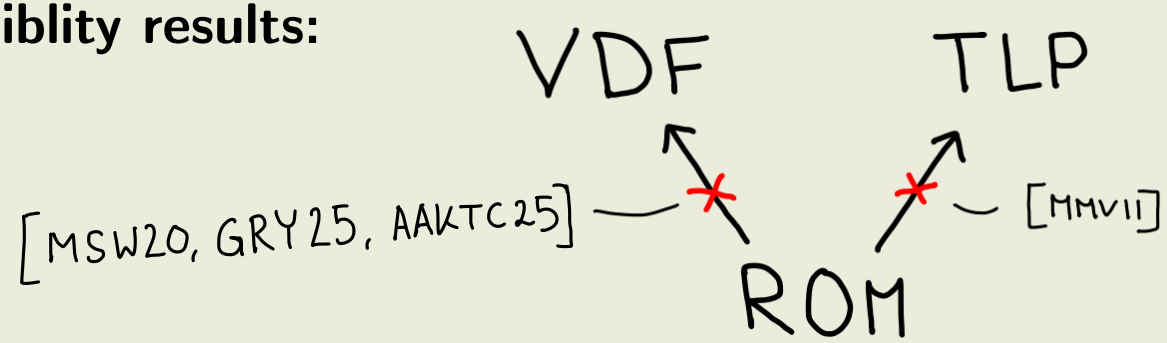


Constructing $\mathcal{M}_{\mathrm{VDF}}$ & $\mathcal{M}_{\mathrm{TLP}}$

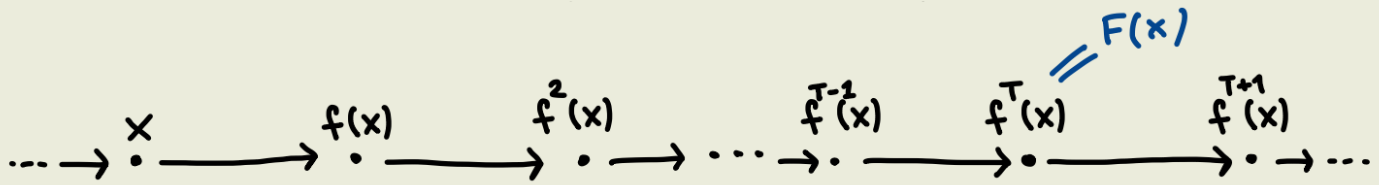
Starting point: (parallel) Random Oracle Model (ROM)

- All algorithms have access to a random function f
- Complexity is measured exclusively by oracle queries

We have impossiblity results:



But: there is a simple delay function $F = f^T$:

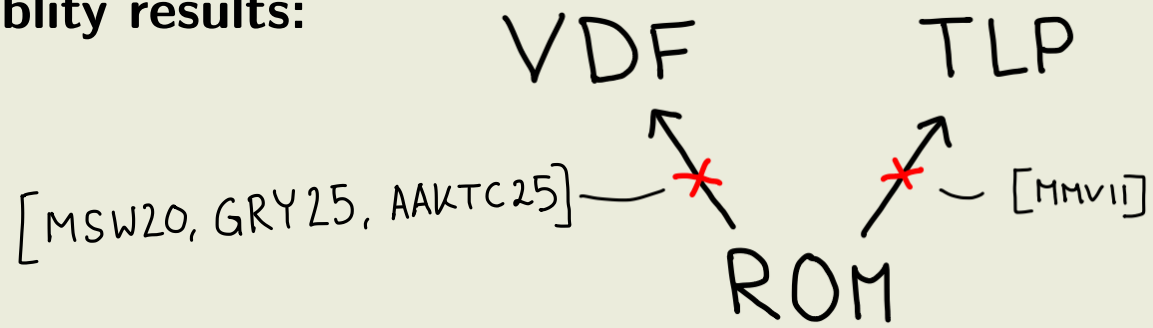


Constructing $\mathcal{M}_{\text{VDF}}$ & $\mathcal{M}_{\text{TLP}}$

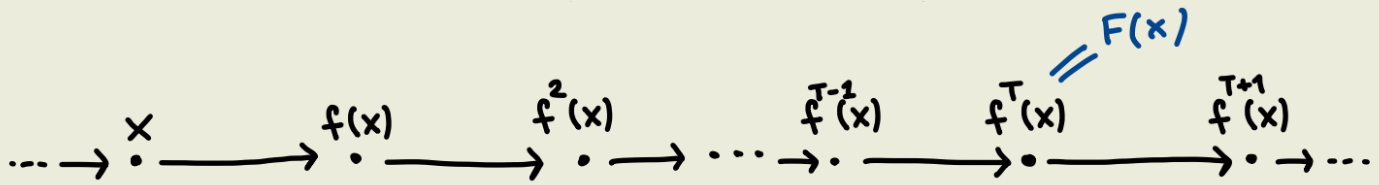
Starting point: (parallel) Random Oracle Model (ROM)

- All algorithms have access to a random function f
- Complexity is measured exclusively by oracle queries

We have impossiblity results:



But: there is a simple delay function $F = f^T$:



Idea: Extend the ROM with minimal additional oracles s.t. we can construct one of the two primitives, but (still) rule out the other.

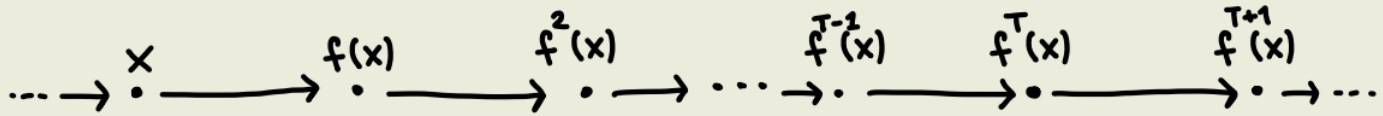
Construction of $\mathcal{M}_{\mathrm{VDF}}$

- Want VDFs to exist, and TLPs to (provably) not exist

Construction of $\mathcal{M}_{\mathrm{VDF}}$

- Want VDFs to exist, and TLPs to (provably) not exist
- **Idea:** Extend the ROM with a verification oracle V_f:

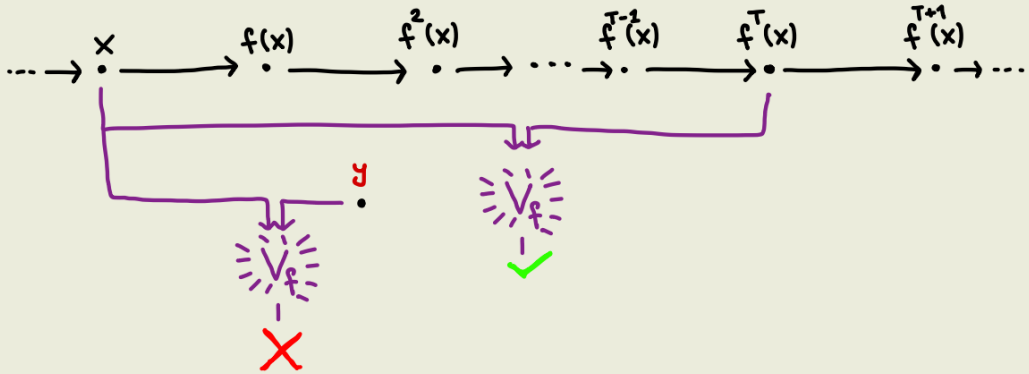
  $V_f(x, y) = 1$ if and only if $f^T(x) = y$.



Construction of $\mathcal{M}_{\mathrm{VDF}}$

- Want VDFs to exist, and TLPs to (provably) not exist
- **Idea:** Extend the ROM with a verification oracle V_f:

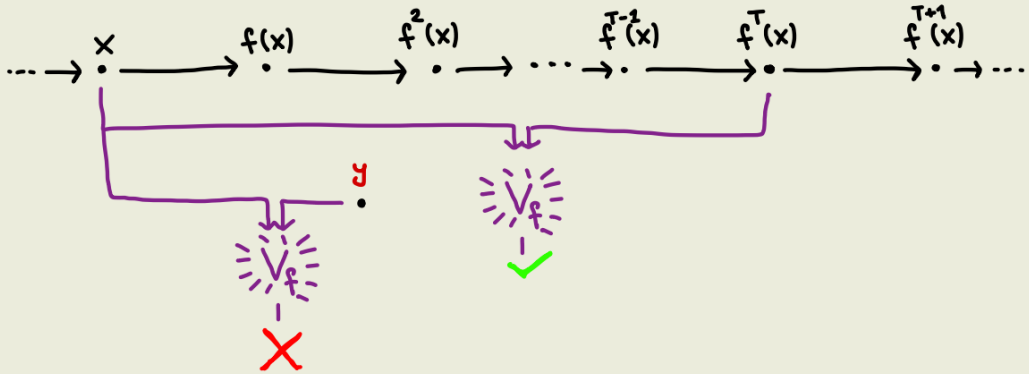
  $V_f(x, y) = 1$ if and only if $f^T(x) = y$.



Construction of $\mathcal{M}_{\mathrm{VDF}}$

- Want VDFs to exist, and TLPs to (provably) not exist
- **Idea:** Extend the ROM with a verification oracle V_f:

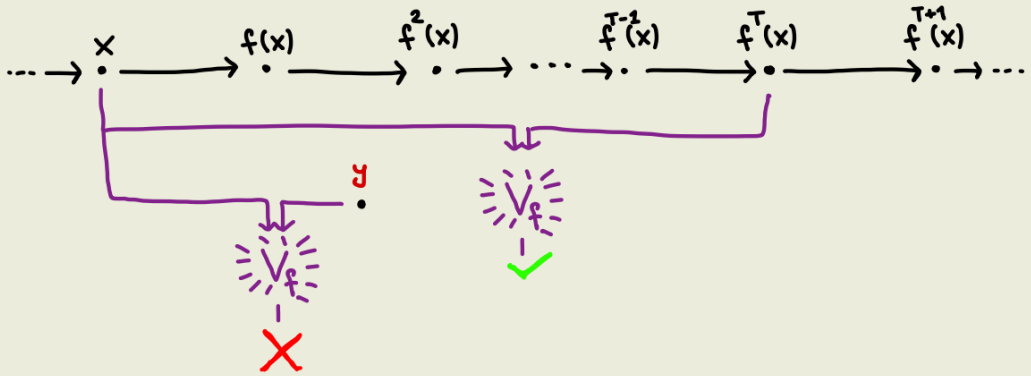
  $V_f(x, y) = 1$ if and only if $f^T(x) = y$.



Construction of $\mathcal{M}_{\mathrm{VDF}}$

- Want VDFs to exist, and TLPs to (provably) not exist
- **Idea:** Extend the ROM with a verification oracle V_f:

  $V_f(x, y) = 1$ if and only if $f^T(x) = y$.

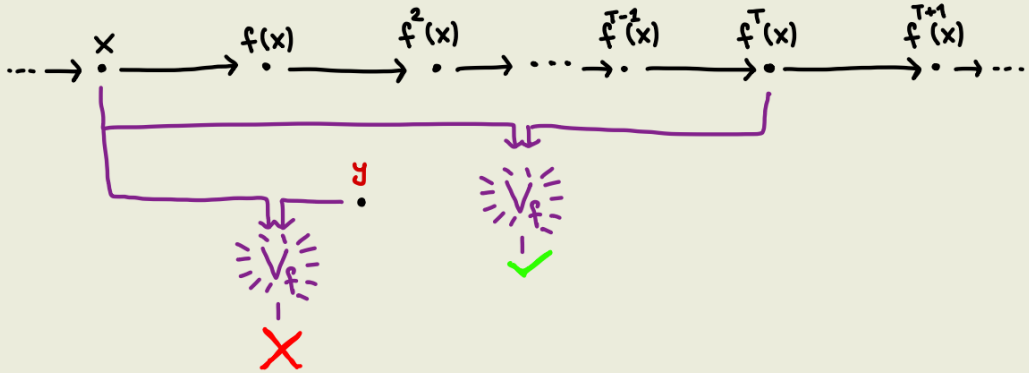


- Now:
  - VDFs exist

Construction of $\mathcal{M}_{\mathrm{VDF}}$

- Want VDFs to exist, and TLPs to (provably) not exist
- **Idea:** Extend the ROM with a verification oracle V_f:

  $V_f(x, y) = 1$ if and only if $f^T(x) = y$.

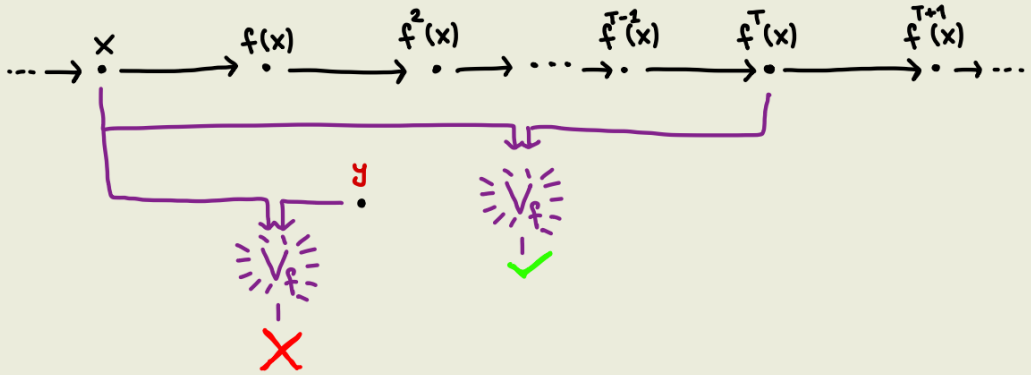


- Now:
  - VDFs exist
  - TLPs can (still) be ruled out, by building on the proof in [MMV11].

Construction of $\mathcal{M}_{\mathrm{VDF}}$

- Want VDFs to exist, and TLPs to (provably) not exist
- **Idea:** Extend the ROM with a verification oracle V_f:

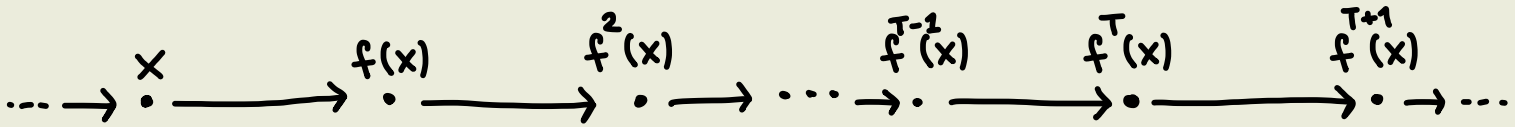
  $V_f(x, y) = 1$ if and only if $f^T(x) = y$.



- Now:
  - VDFs exist
  - TLPs can (still) be ruled out, by building on the proof in [MMV11].
  - **Key Insight:** We show that given a TLP in this model we can construct a TLP in the plain ROM by simulting the verification oracle V_f using some extra queries.

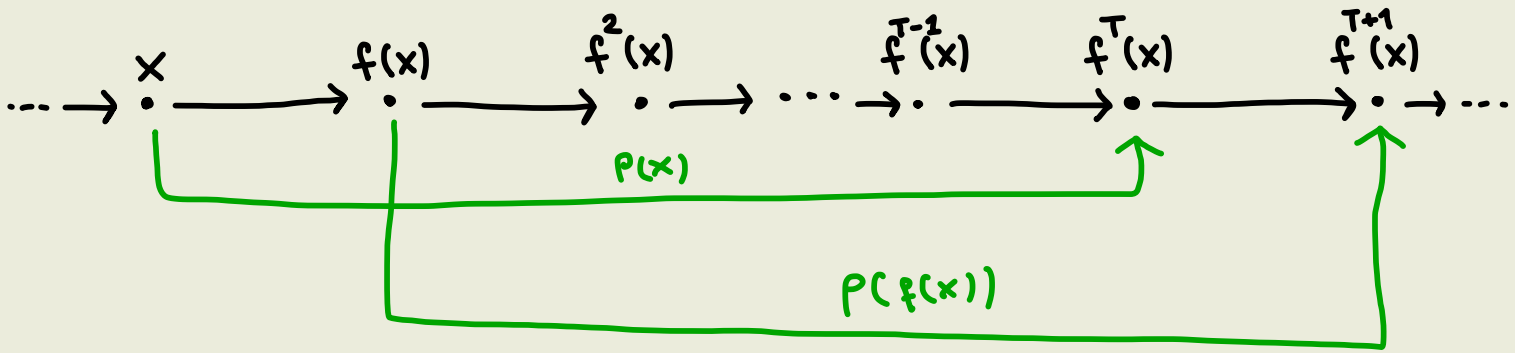
Construction of $\mathcal{M}_{\mathrm{TLP}}$

Want TLPs to exist, but VDFs to (provably) not exist.



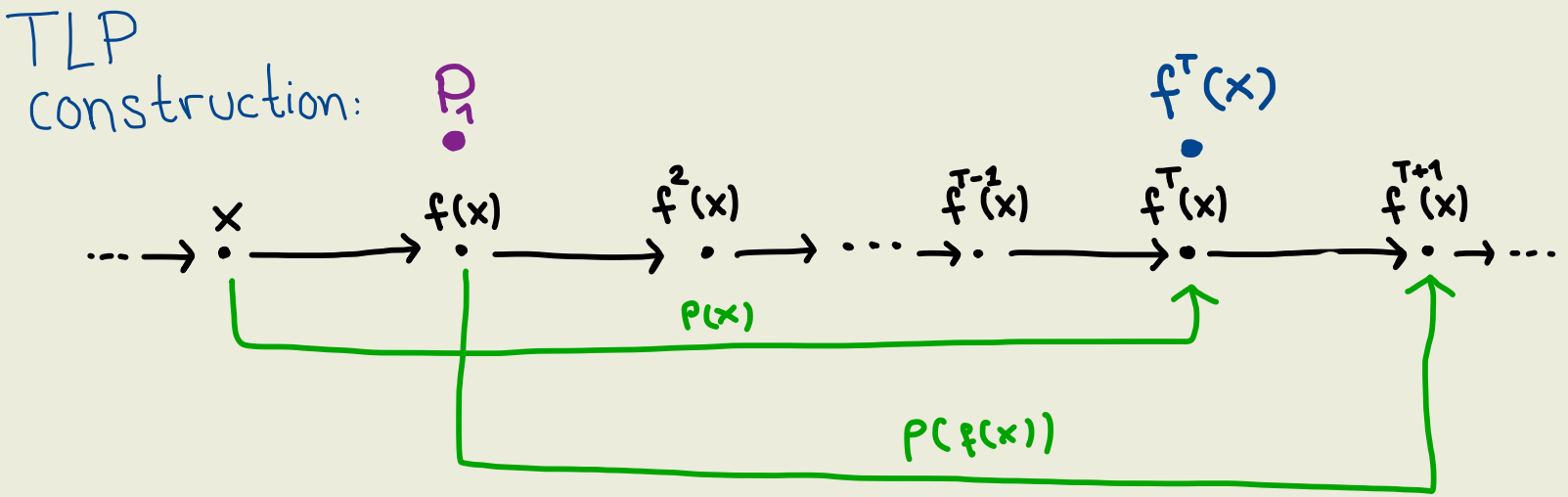
Construction of $\mathcal{M}_{\mathrm{TLP}}$

Want TLPs to exist, but VDFs to (provably) not exist.
Naive Idea: Extend the ROM with "puzzle oracle": $\rho(x) := f^T(x)$.



Construction of $\mathcal{M}_{\mathrm{TLP}}$

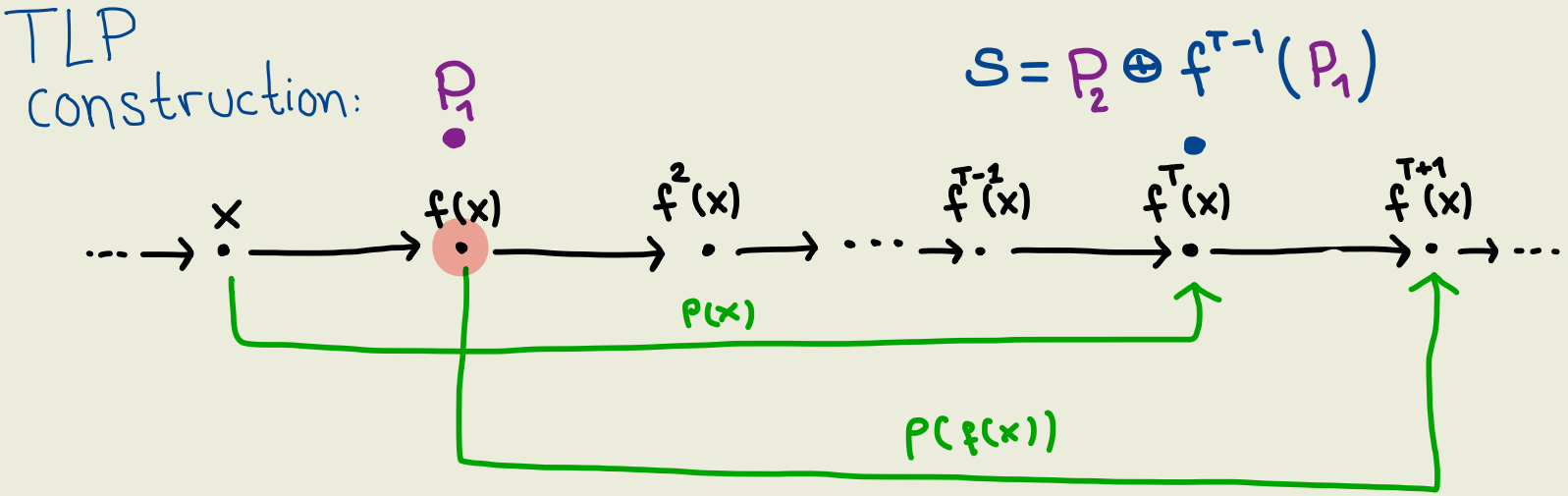
Want TLPs to exist, but VDFs to (provably) not exist.
Naive Idea: Extend the ROM with "puzzle oracle": $\rho(x) := f^T(x)$.



- TLPs exist.
  - PGen(s, T) outputs $(p_1, p_2) = (f(x), \rho(x) \oplus s)$ for $x \leftarrow \{0,1\}^\lambda$

Construction of $\mathcal{M}_{\mathrm{TLP}}$

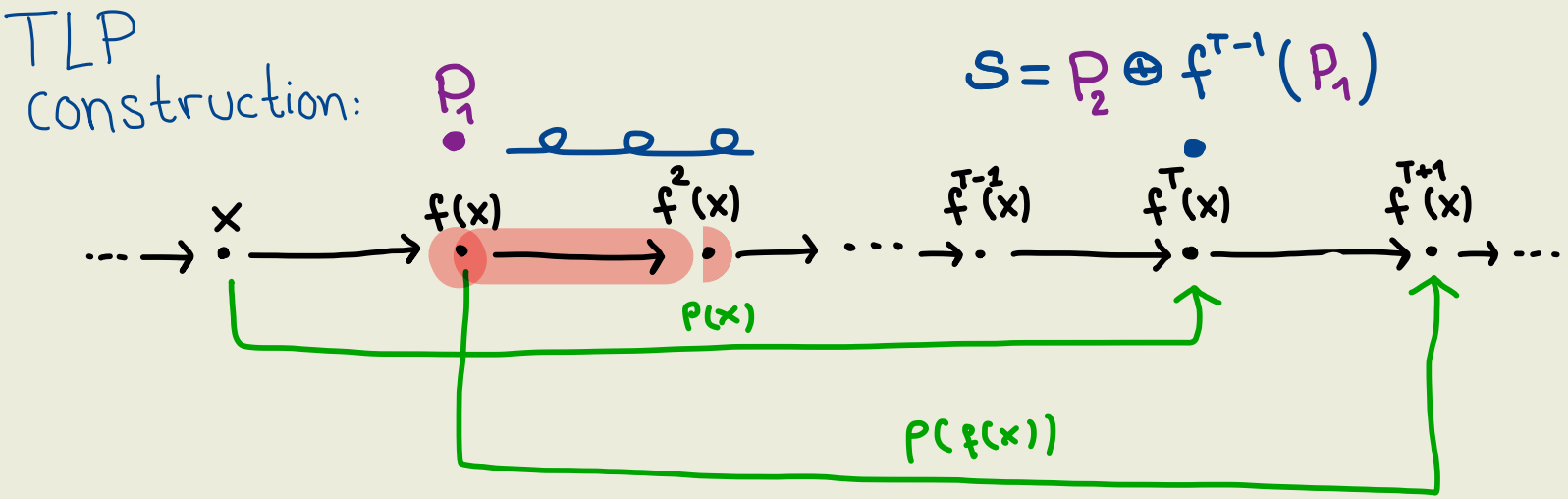
Want TLPs to exist, but VDFs to (provably) not exist.
Naive Idea: Extend the ROM with "puzzle oracle": $\rho(x) := f^T(x)$.



- TLPs exist.
  - PGen(s, T) outputs $(p_1, p_2) = (f(x), \rho(x) \oplus s)$ for $x \leftarrow \{0,1\}^\lambda$
  - Sol(p_1, p_2) outputs $f^{T-1}(p_1) \oplus p_2$.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

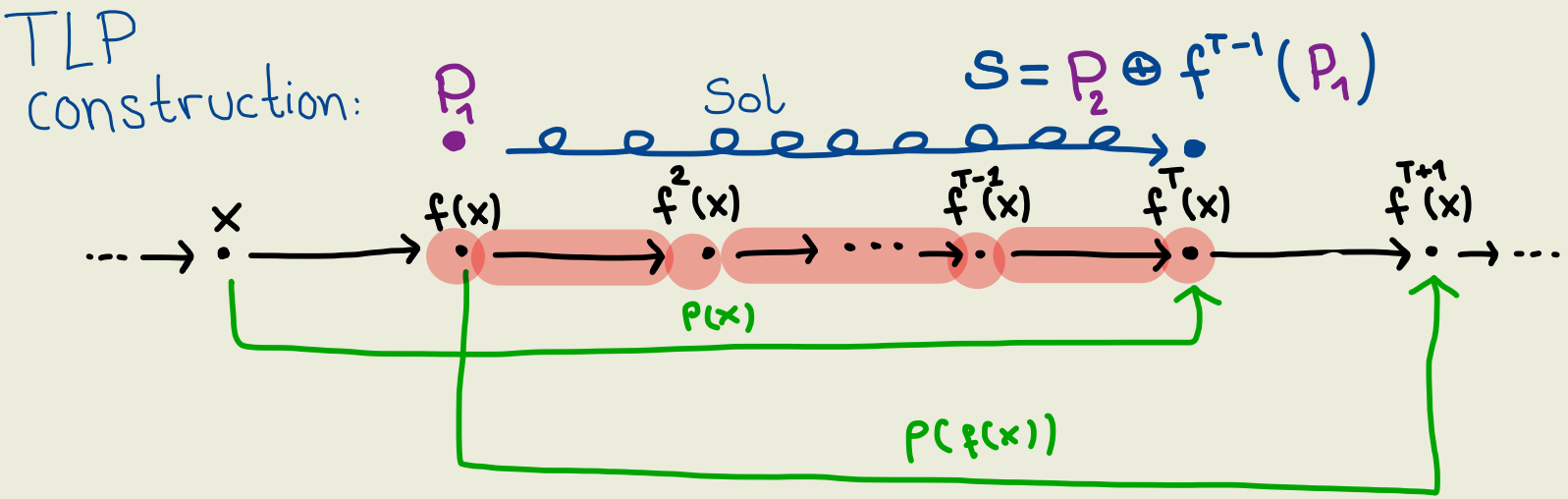
Want TLPs to exist, but VDFs to (provably) not exist.
Naive Idea: Extend the ROM with "puzzle oracle": $\rho(x) := f^T(x)$.



- TLPs exist.
  - $\mathsf{PGen}(s, T)$ outputs $(p_1, p_2) = (f(x), \rho(x) \oplus s)$ for $x \leftarrow \{0,1\}^\lambda$
  - $\mathsf{Sol}(p_1, p_2)$ outputs $f^{T-1}(p_1) \oplus p_2$.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

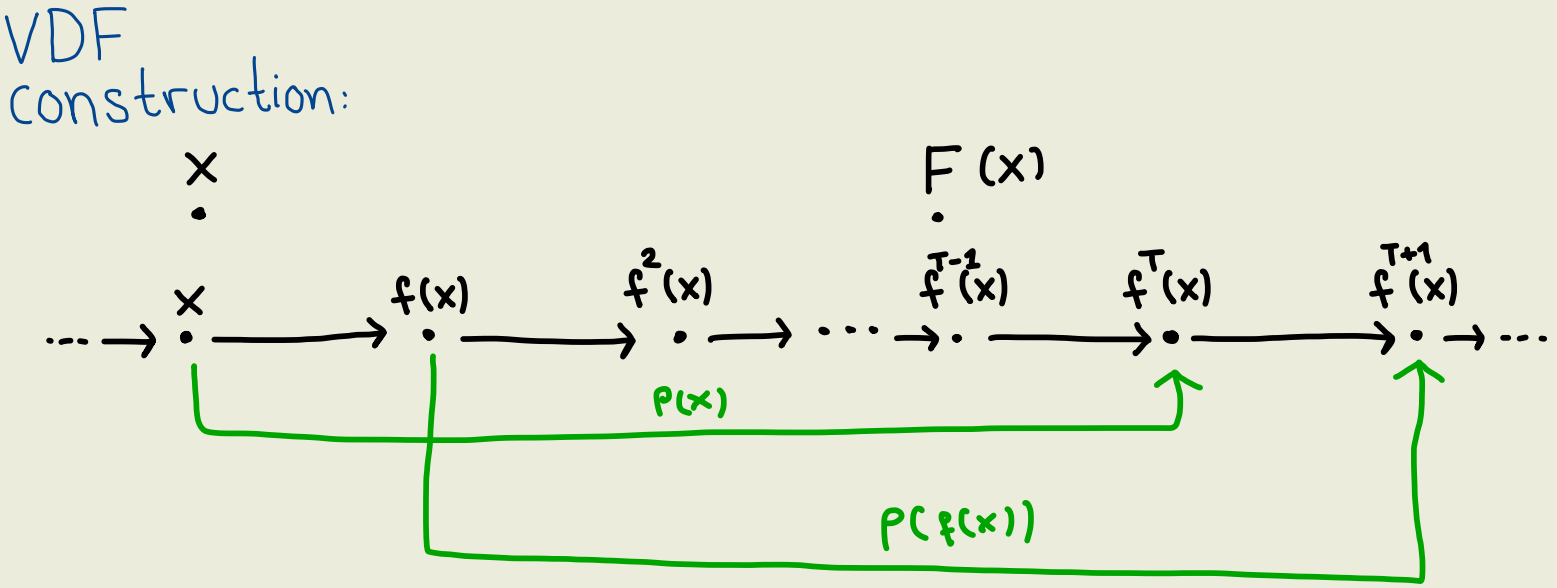
Want TLPs to exist, but VDFs to (provably) not exist.
Naive Idea: Extend the ROM with "puzzle oracle": $\rho(x) := f^T(x)$.



- TLPs exist.
  - PGen(s, T) outputs $(p_1, p_2) = (f(x), \rho(x) \oplus s)$ for $x \leftarrow \{0,1\}^\lambda$
  - Sol(p_1, p_2) outputs $f^{T-1}(p_1) \oplus p_2$.

Construction of $\mathcal{M}_{\text{TLP}}$

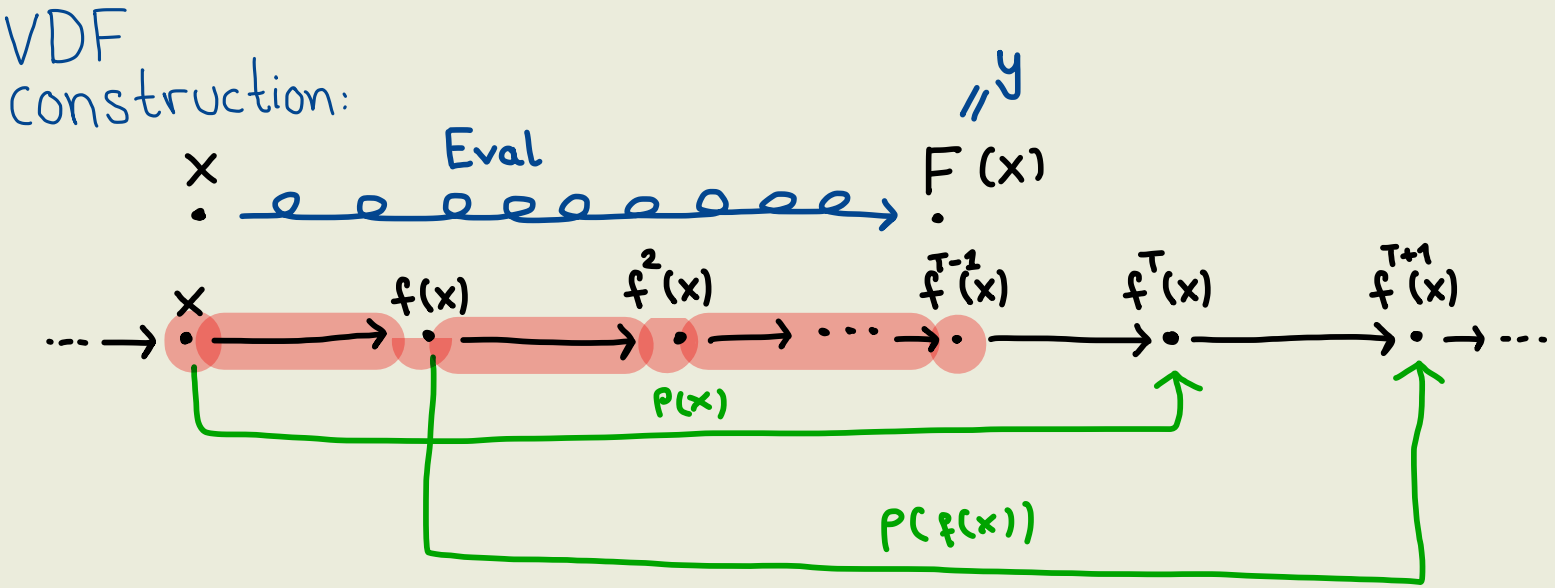
Want TLPs to exist, but VDFs to (provably) not exist.
Naive Idea: Extend the ROM with "puzzle oracle": $\rho(x) := f^T(x)$.



- TLPs exist.
- But: VDFs exist as well!
  - given $x \in \{0,1\}^\lambda$, Eval(x) computes $F(x) := f^{T-1}(x)$ (no proof)
  - given x, y, Verify checks if $f(y) = \rho(x) = f^T(x)$.

Construction of $\mathcal{M}_{\text{TLP}}$

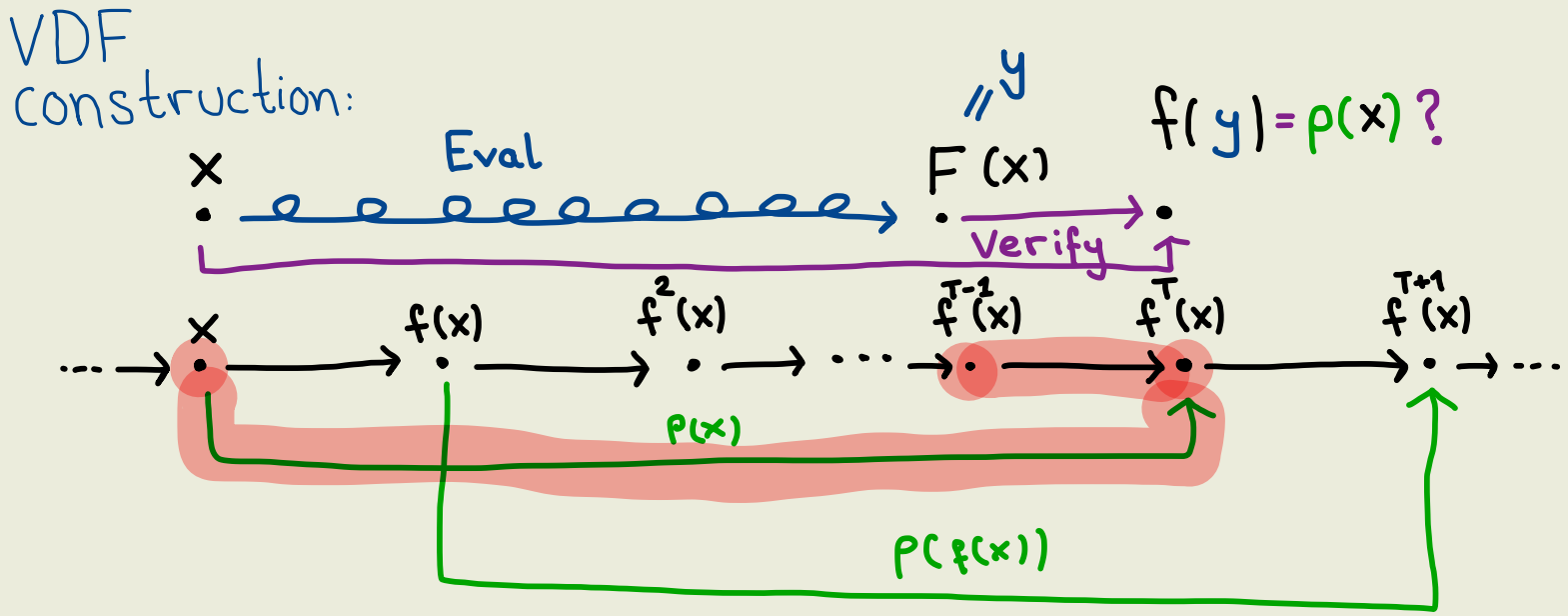
Want TLPs to exist, but VDFs to (provably) not exist.
Naive Idea: Extend the ROM with "puzzle oracle": $\rho(x) := f^T(x)$.



- TLPs exist.
- But: VDFs exist as well!
  - given $x \in \{0,1\}^\lambda$, Eval(x) computes $F(x) := f^{T-1}(x)$ (no proof)
  - given x, y, Verify checks if $f(y) = \rho(x) = f^T(x)$.

Construction of $\mathcal{M}_{\text{TLP}}$

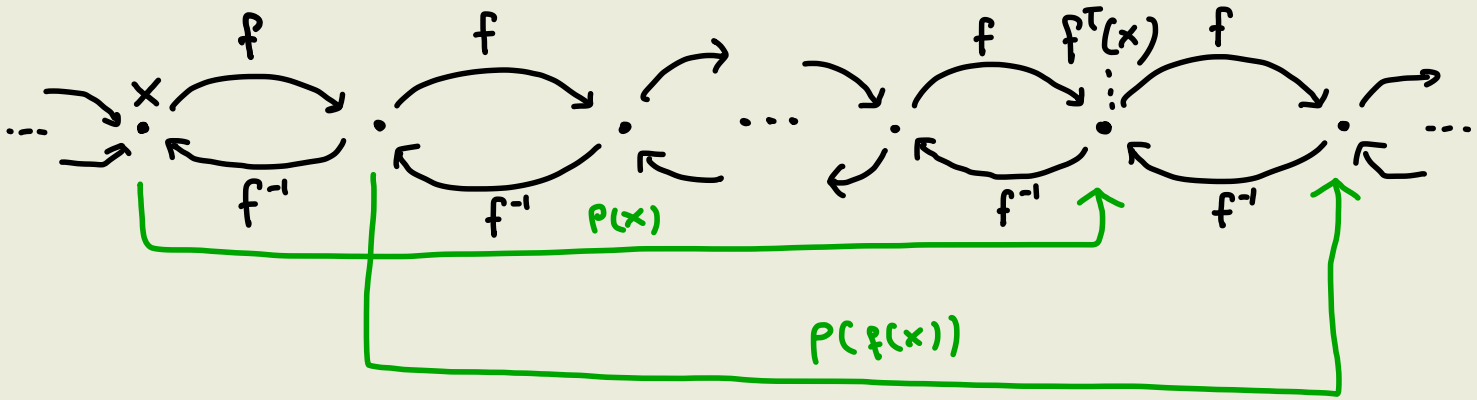
Want TLPs to exist, but VDFs to (provably) not exist.
Naive Idea: Extend the ROM with "puzzle oracle": $\rho(x) := f^T(x)$.



- TLPs exist.
- But: VDFs exist as well!
  - given $x \in \{0,1\}^\lambda$, Eval(x) computes $F(x) := f^{T-1}(x)$ (no proof)
  - given x, y, Verify checks if $f(y) = \rho(x) = f^T(x)$.

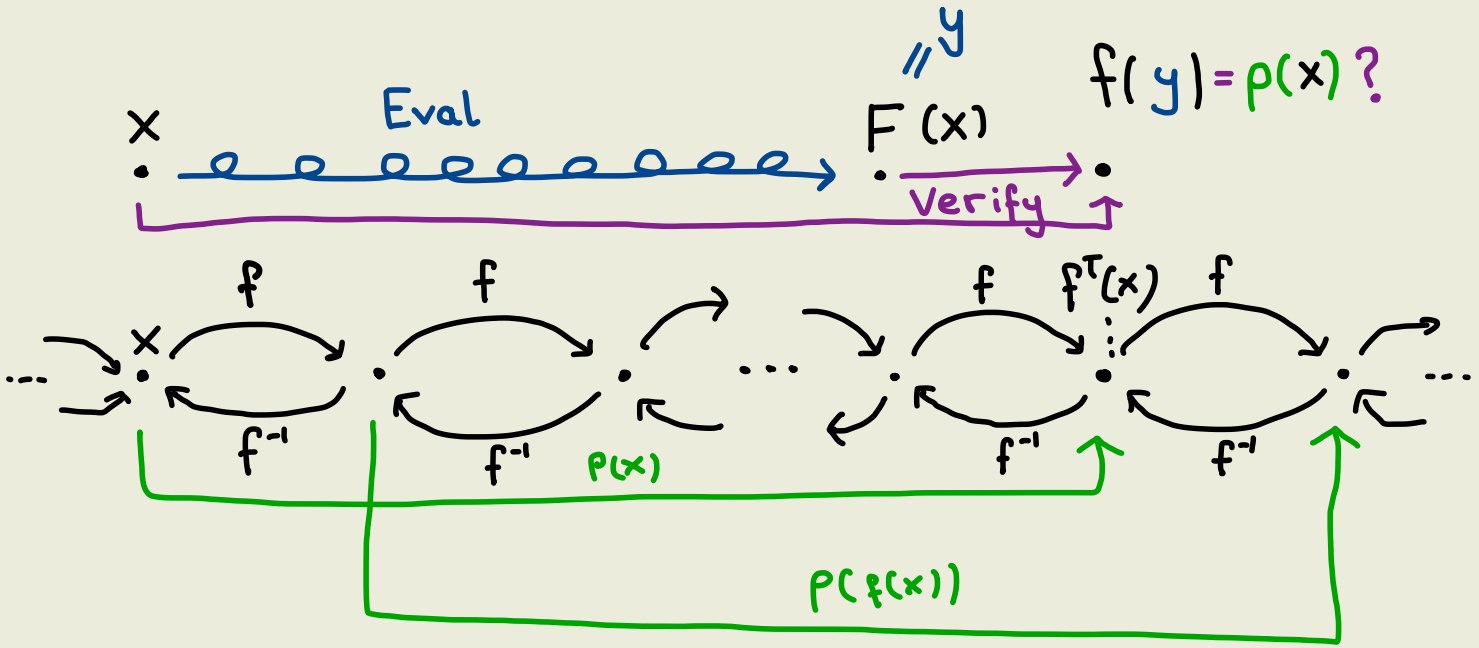
Construction of $\mathcal{M}_{\mathrm{TLP}}$

Better Idea: Let f be a random permutation, and provide access to f^{-1}!



Construction of $\mathcal{M}_{\mathrm{TLP}}$

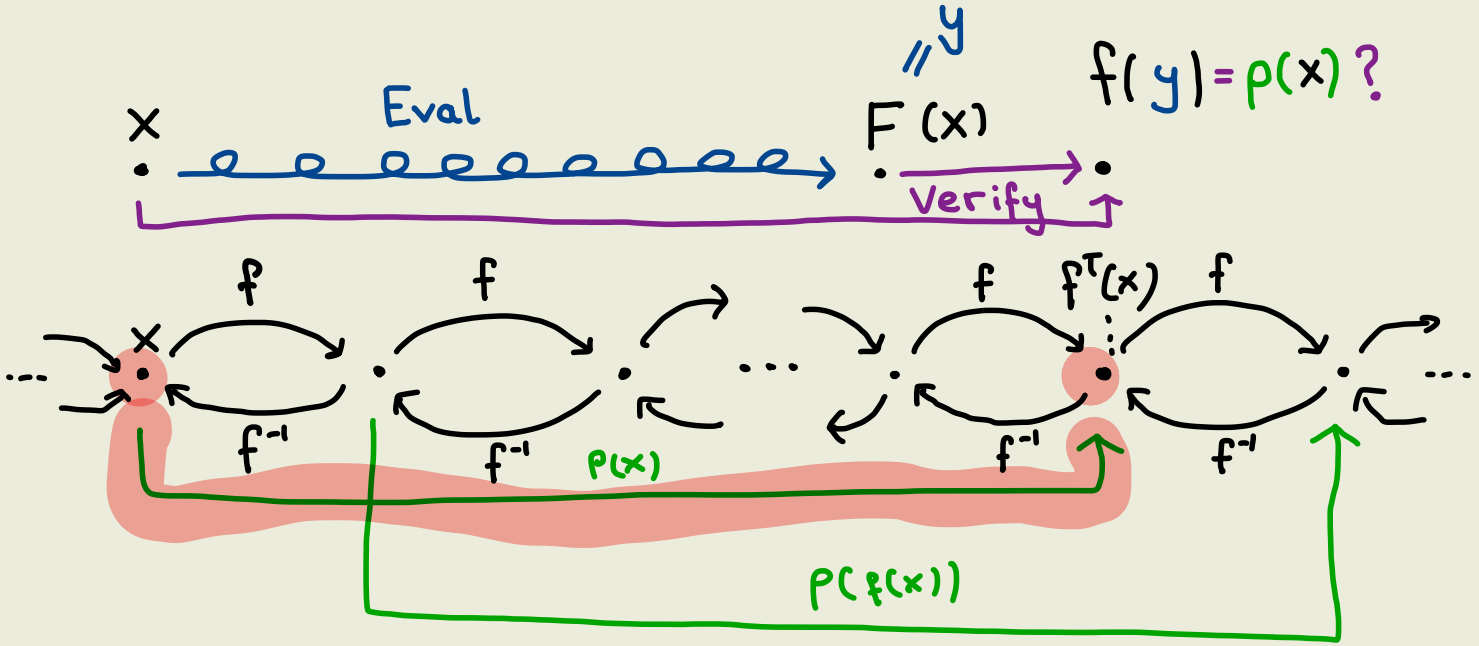
Better Idea: Let f be a random permutation, and provide access to f^{-1}!



- Now: the construction of VDF as above is insecure: $F(x)$ can be computed in 2 steps.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

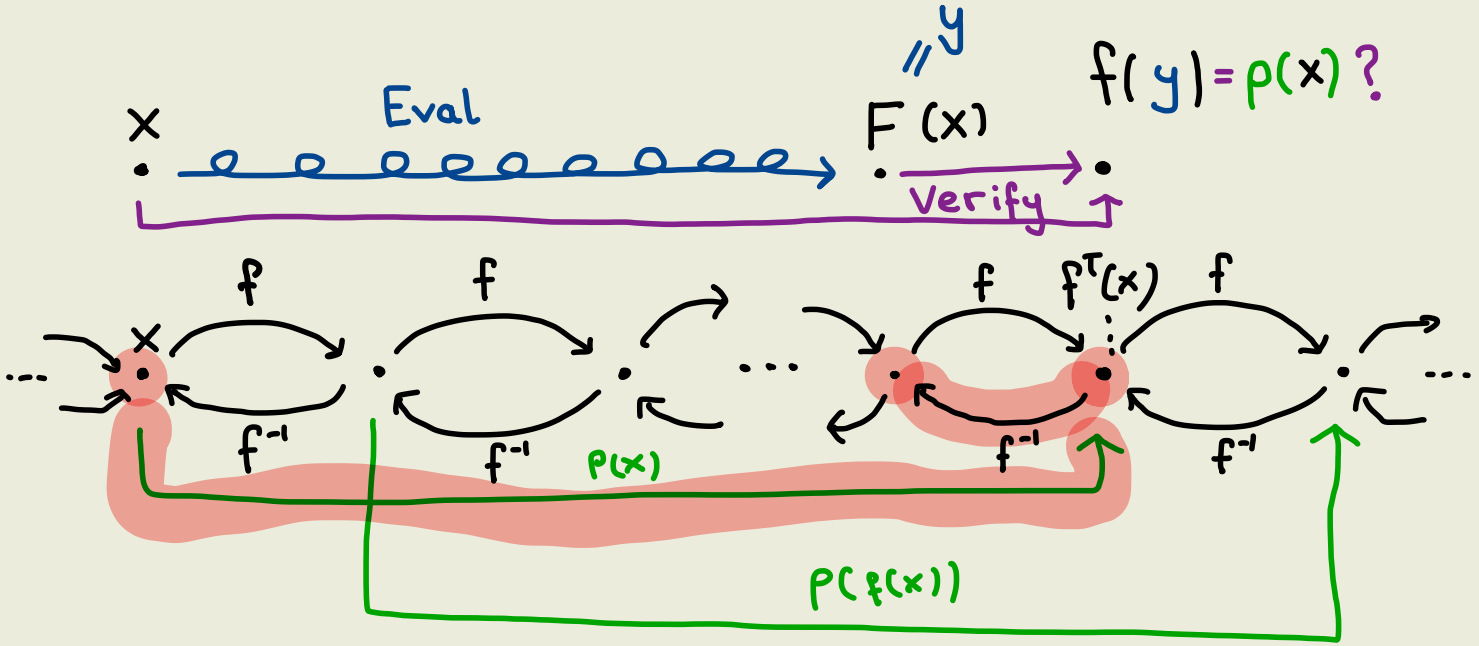
Better Idea: Let f be a random permutation, and provide access to f^{-1}!



- Now: the construction of VDF as above is insecure: $F(x)$ can be computed in 2 steps.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

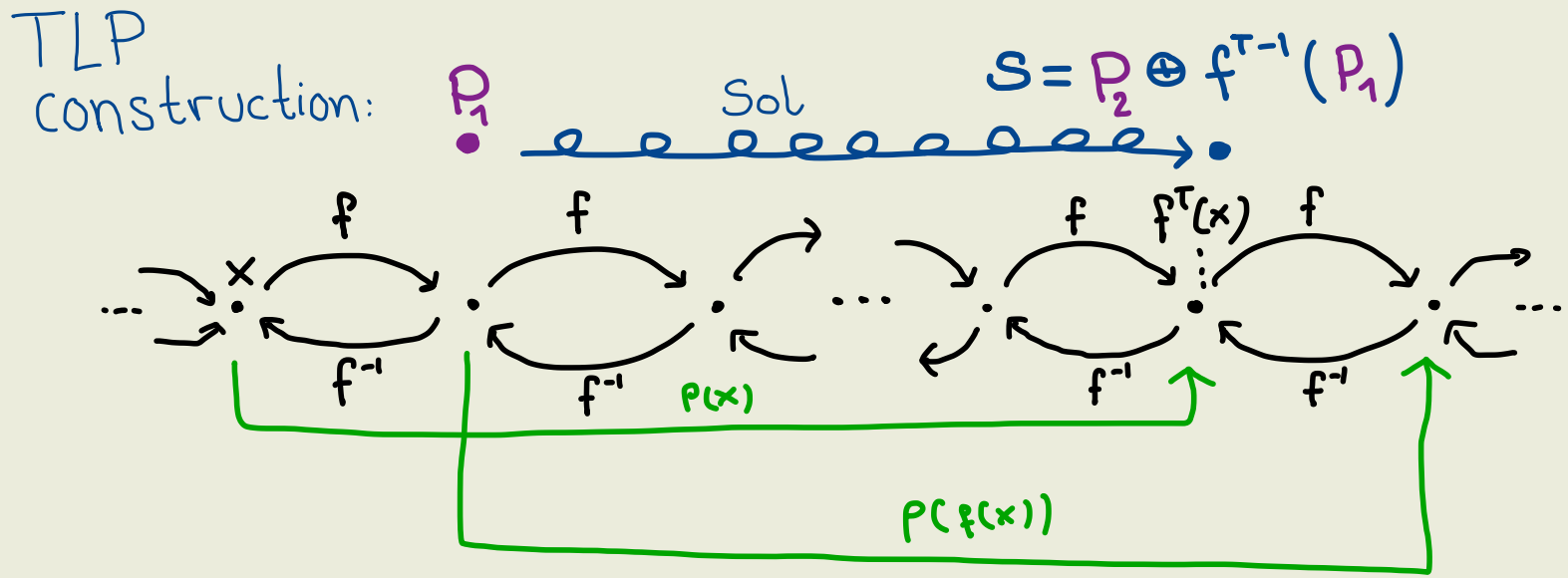
Better Idea: Let f be a random permutation, and provide access to f^{-1}!



- Now: the construction of VDF as above is insecure: $F(x)$ can be computed in 2 steps.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

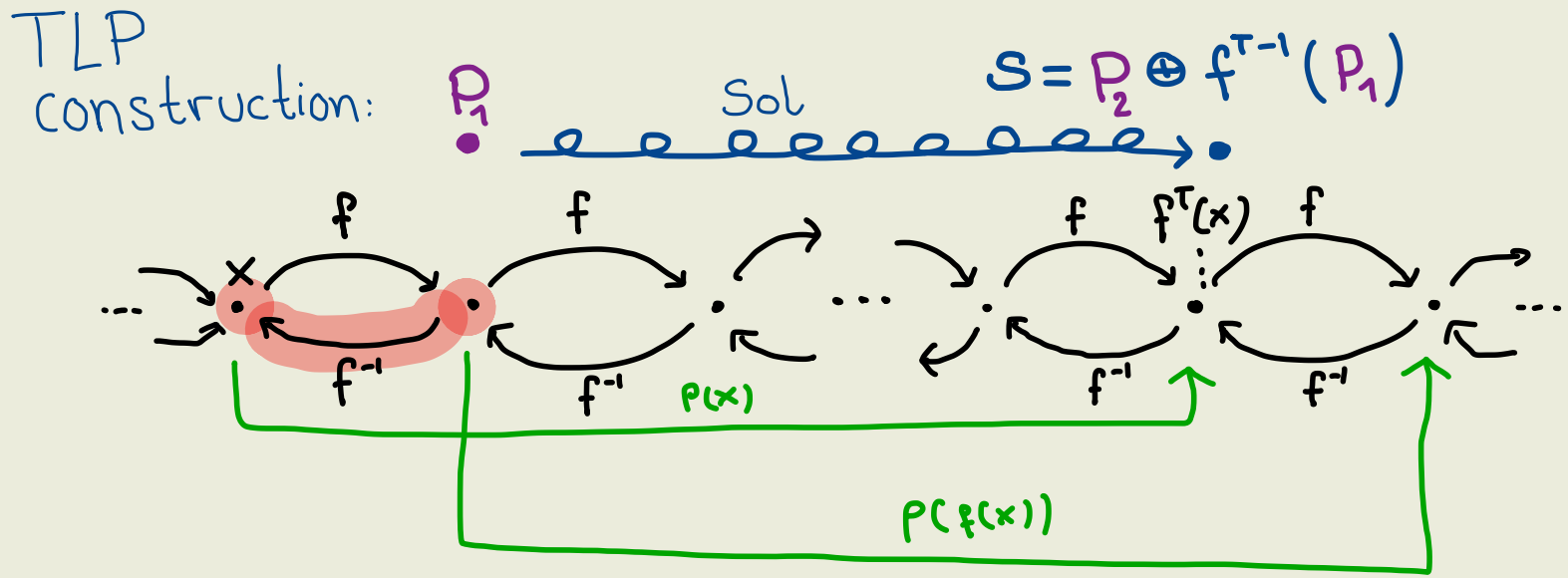
Better Idea: Let f be a random permutation, and provide access to f^{-1}!



- Now: the construction of VDF as above is insecure: $F(x)$ can be computed in 2 steps.
- But: the construction of TLP as above is also insecure.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

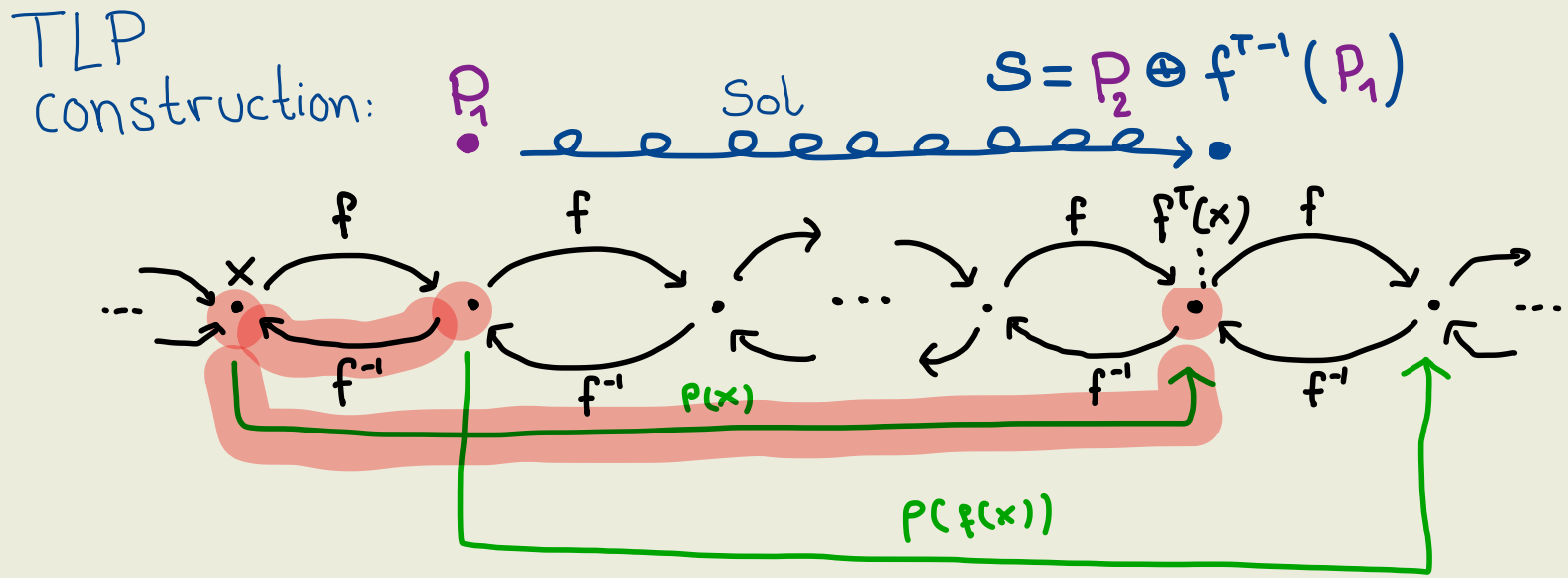
Better Idea: Let f be a random permutation, and provide access to f^{-1}!



- Now: the construction of VDF as above is insecure: $F(x)$ can be computed in 2 steps.
- But: the construction of TLP as above is also insecure.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

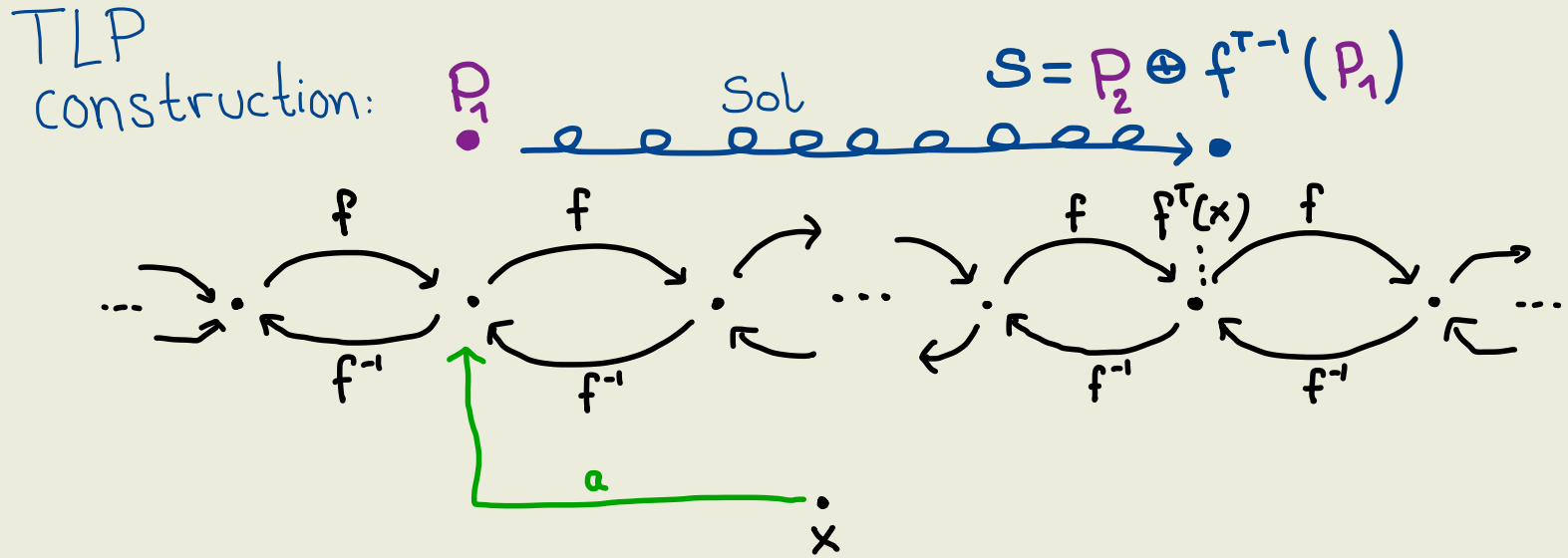
Better Idea: Let f be a random permutation, and provide access to f^{-1}!



- Now: the construction of VDF as above is insecure: $F(x)$ can be computed in 2 steps.
- But: the construction of TLP as above is also insecure.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

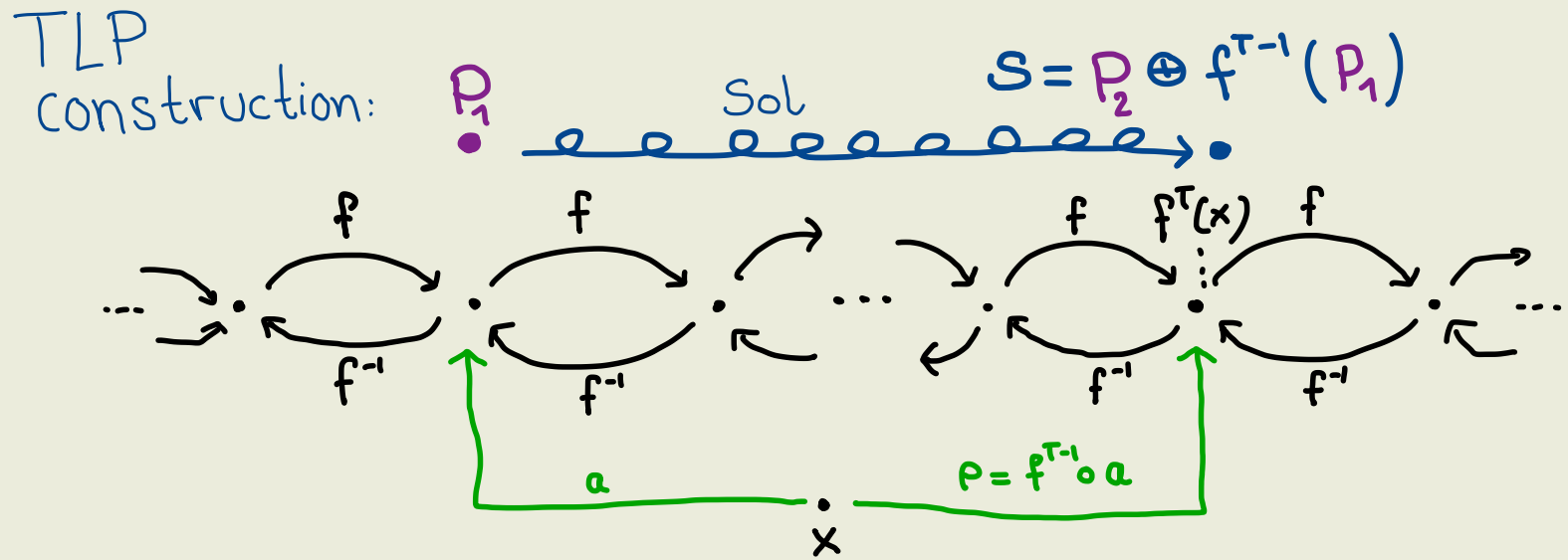
Better Idea: Let f be a random permutation, and provide access to f^{-1}!



- Now: the construction of VDF as above is insecure: $F(x)$ can be computed in 2 steps.
- But: the construction of TLP as above is also insecure.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

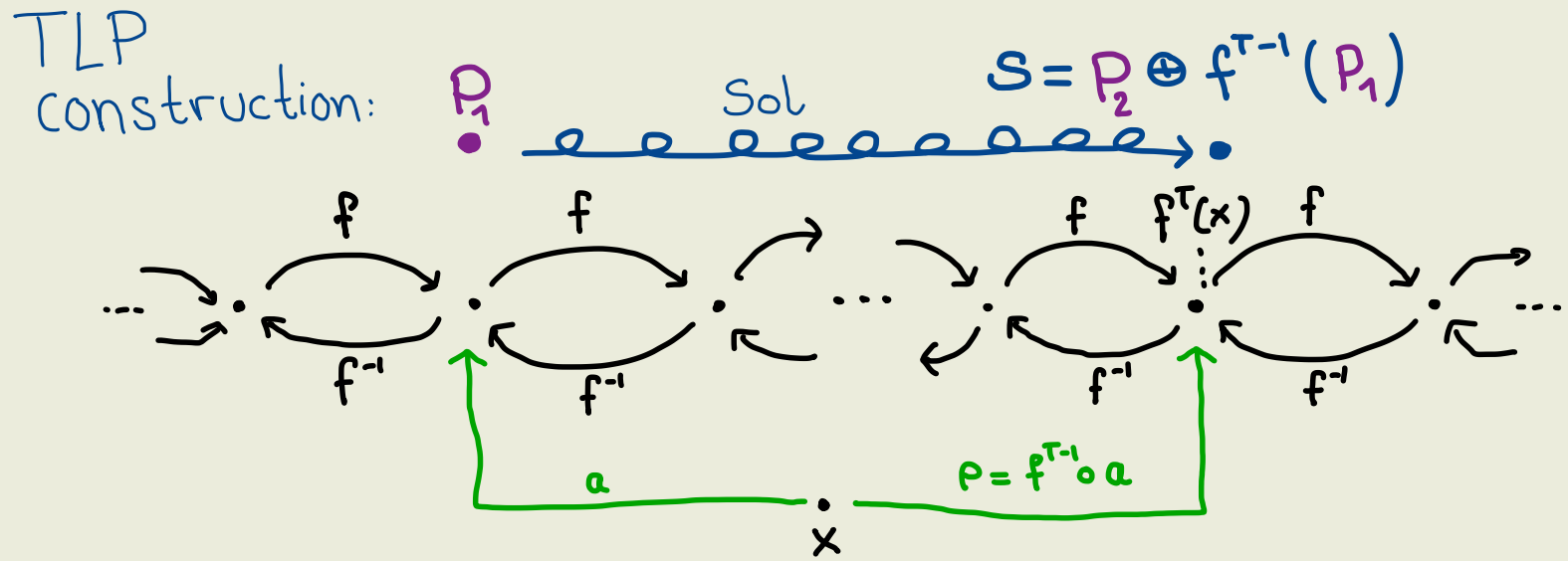
Better Idea: Let f be a random permutation, and provide access to f^{-1}!



- Now: the construction of VDF as above is insecure: $F(x)$ can be computed in 2 steps.
- But: the construction of TLP as above is also insecure.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

Better Idea: Let f be a random permutation, and provide access to f^{-1}!



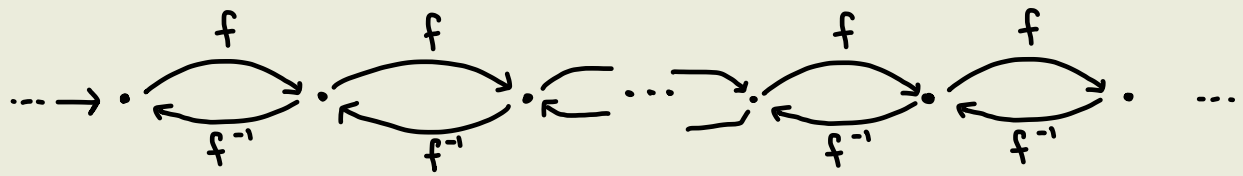
- ~~Now: the construction of VDF as above is insecure: $F(x)$ can be computed in 2 steps.~~
- ~~But: the construction of TLP as above is also insecure.~~

Construction of $\mathcal{M}_{\mathrm{TLP}}$

Final Construction:

- $f\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a random permutation

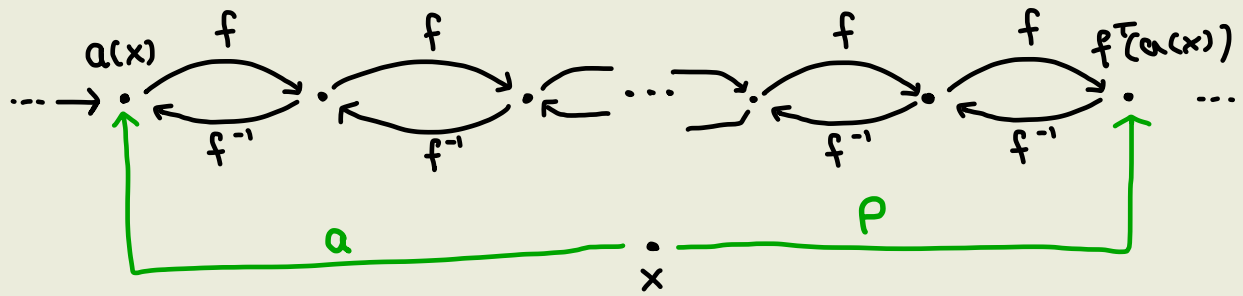
- In $\mathcal{M}_{\mathrm{TLP}}$, all algorithms have shared access to:
  - f and its inverse f^{-1}, and additionaly



Construction of $\mathcal{M}_{\mathrm{TLP}}$

Final Construction:

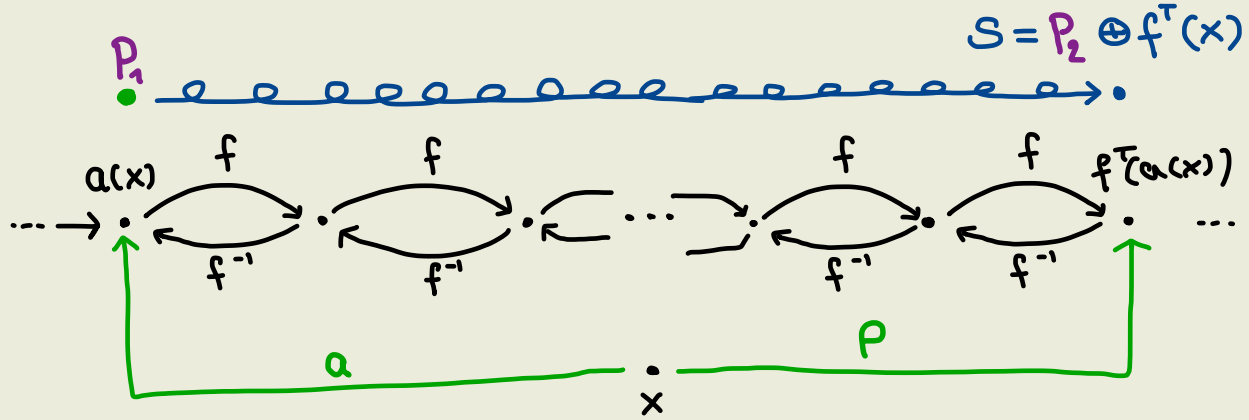
- $f\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a random permutation
- $a\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a random function
- In $\mathcal{M}_{\mathrm{TLP}}$, all algorithms have shared access to:
  - f and its inverse f^{-1}, and additionaly
  - the maps $x \mapsto a(x)$ and $x \mapsto \rho(x) := f^T(a(x))$ ("puzzle oracles")



Construction of $\mathcal{M}_{\mathrm{TLP}}$

Final Construction:

- $f\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a random permutation
- $a\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a random function
- In $\mathcal{M}_{\mathrm{TLP}}$, all algorithms have shared access to:
  - f and its inverse f^{-1}, and additionaly
  - the maps $x \mapsto a(x)$ and $x \mapsto \rho(x) := f^T(a(x))$ ("puzzle oracles")



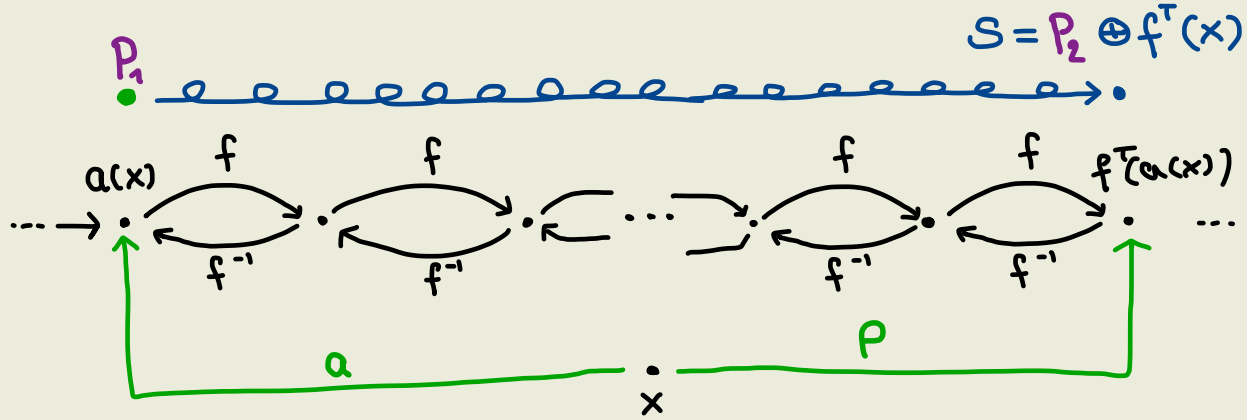
Now:

- TLPs exist: $\mathrm{PGen}(s,T)$ outputs $(a(x), \rho(x) \oplus s)$ for $x \leftarrow \{0,1\}^\lambda$.

Construction of $\mathcal{M}_{\mathrm{TLP}}$

Final Construction:

- $f\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a random permutation
- $a\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a random function
- In $\mathcal{M}_{\mathrm{TLP}}$, all algorithms have shared access to:
  - f and its inverse f^{-1}, and additionaly
  - the maps $x \mapsto a(x)$ and $x \mapsto \rho(x) := f^T(a(x))$ ("puzzle oracles")



Now:

- TLPs exist: $\mathsf{PGen}(s,T)$ outputs $(a(x), \rho(x) \oplus s)$ for $x \leftarrow \{0,1\}^\lambda$.
- VDFs don't exist: the proof of [AAKTC25] can be made work in $\mathcal{M}_{\mathrm{TLP}}$. (with lots of work)

Thank you!

Questions?

References I

- Abusalah, Hamza et al. *Impossibility of VDFs in the ROM: The Complete Picture*. Cryptology ePrint Archive, Report 2025/1773. URL: https://eprint.iacr.org/2025/1773.
- Abusalah, Hamza et al. *On Verifiable Delay Functions from Time-Lock Puzzles*. Cryptology ePrint Archive, Report 2025/1782. URL: https://eprint.iacr.org/2025/1782.
- Guan, Ziyi, Artur Riazanov, and Weiqiang Yuan. "Breaking Verifiable Delay Functions in the Random Oracle Model". In: *CRYPTO 2025, Part VII*, pp. 161–191.
- Mahmoody, Mohammad, Tal Moran, and Salil P. Vadhan. "Time-Lock Puzzles in the Random Oracle Model". In: *CRYPTO 2011*, pp. 39–50.
- Mahmoody, Mohammad, Caleb Smith, and David J. Wu. "Can Verifiable Delay Functions Be Based on Random Oracles?" In: *ICALP 2020*, 83:1–83:17.
- Renawi, Omar. *Time-Lock Puzzles from Verifiable Delay Functions and Witness Encryption*. Bachelor Thesis (2020), University of Saarland, Germany. Last accessed on Feb 18, 2026 at https://doi.org/10.60882/cispa.30866072. URL: https://doi.org/10.60882/cispa.30866072.