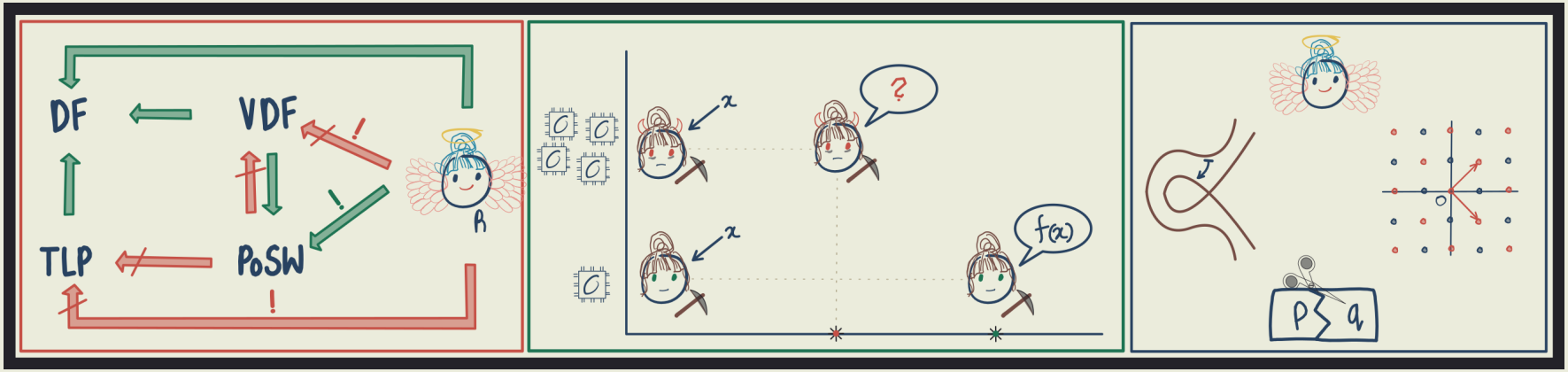


Timed Cryptography: Attacks and Separations

Chethan Kamath, IIT Bombay & Trust Lab



Based on joint works with Hamza Abusalah, Nivesh Aggarwal, Karen Azari, Erkan Tairi & Max von Consbruch

aRtCS Workshop, 09-13/Nov/26

Plan for Today's Talk



Timed Cryptography



Current Landscape



Breaking Random-Oracle Based VDF

Plan for Today's Talk



Timed Cryptography

Current Landscape

Breaking Random-Oracle Based VDF

What is a Timed Primitive?..

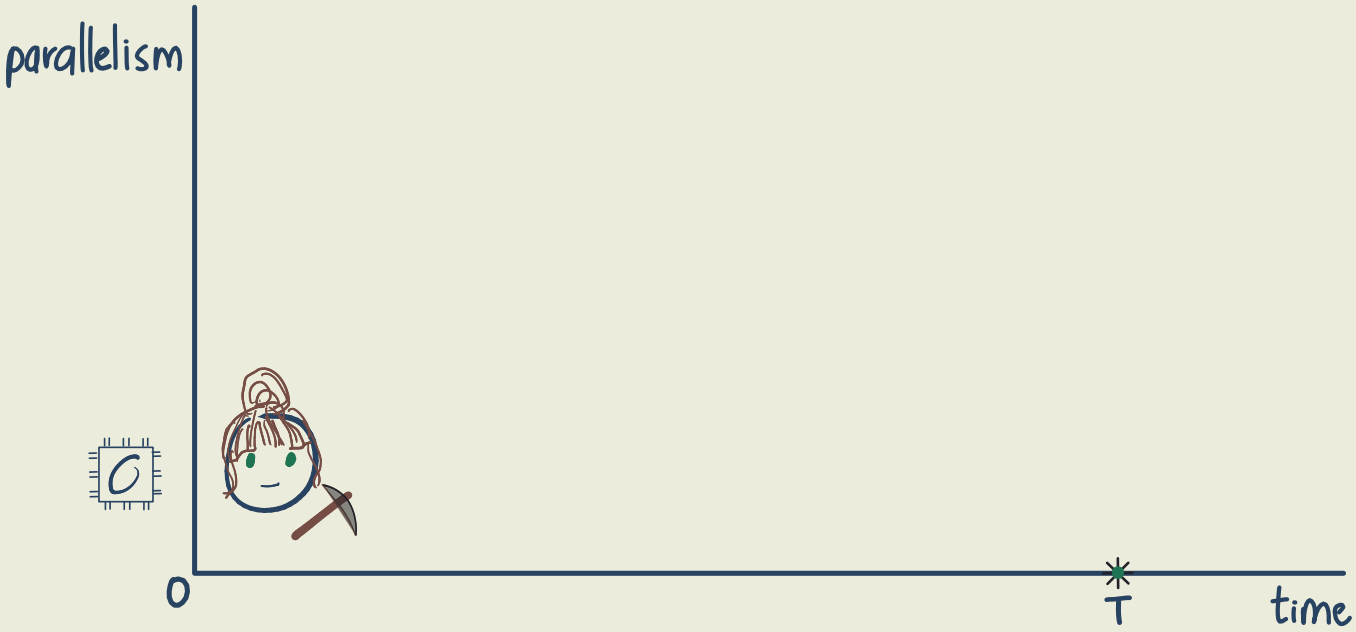


Primitives that deal in “wall-time”

What is a Timed Primitive?..



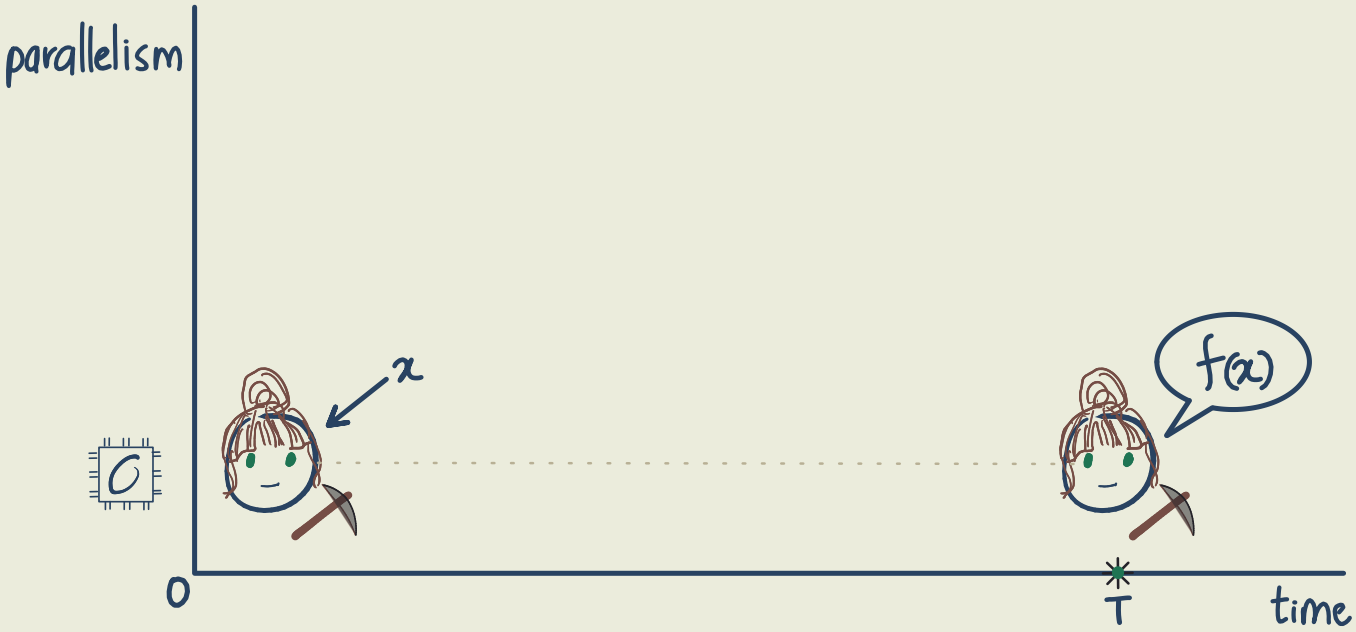
Primitives that deal in "wall-time"



- Simplest example: delay function (DF) $f : \mathcal{X} \to \mathcal{Y}$

What is a Timed Primitive?..

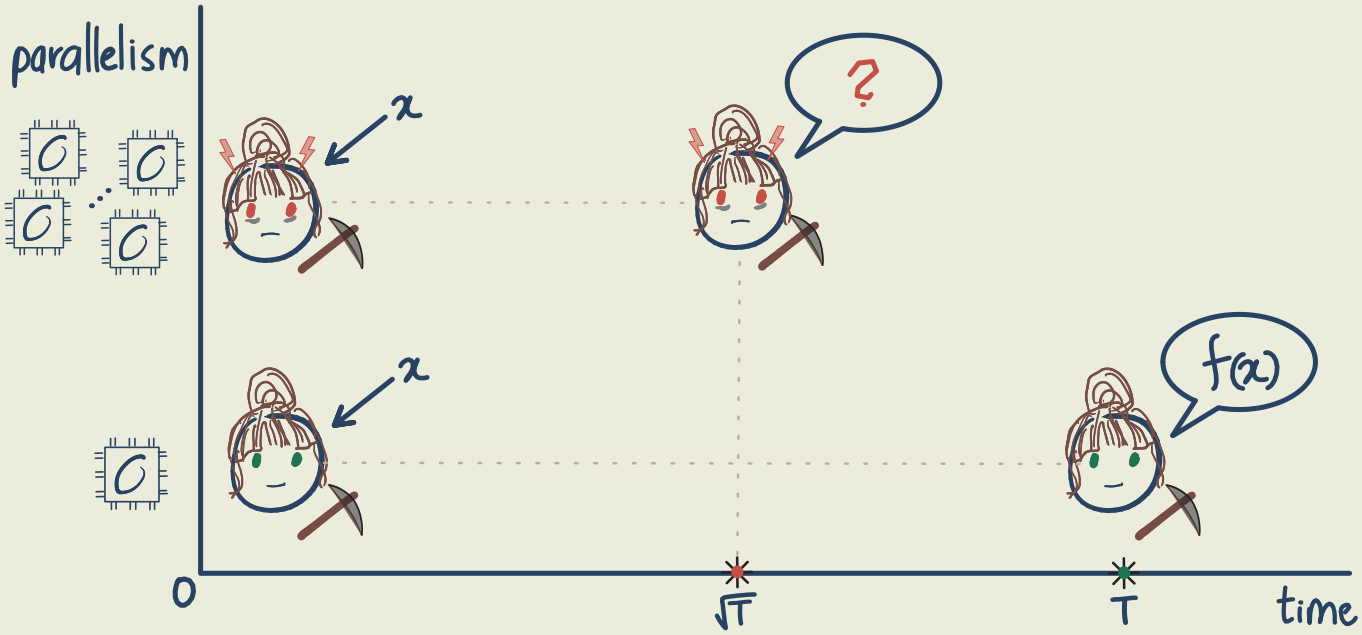
Primitives that deal in "wall-time"



- Simplest example: delay function (DF) $f : \mathcal{X} \to \mathcal{Y}$
  - i) Can be evaluated in time T

What is a Timed Primitive?..

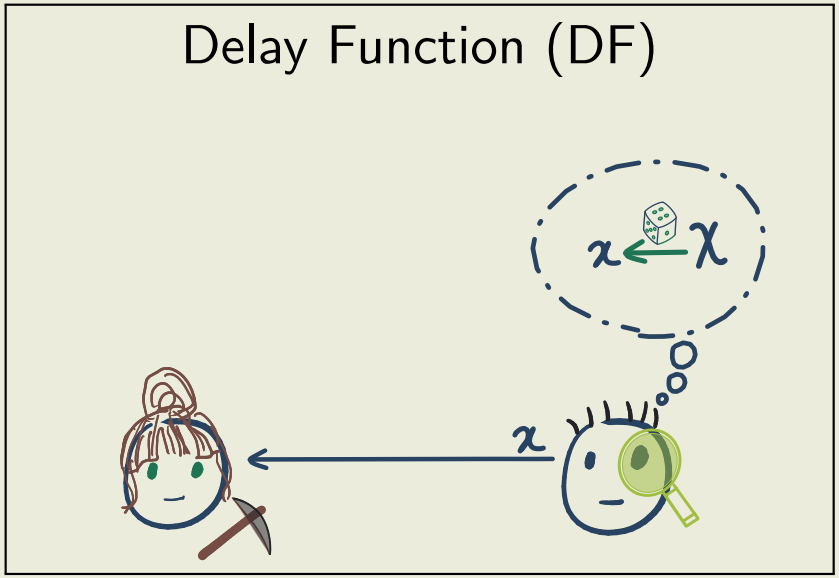
Primitives that deal in "wall-time"



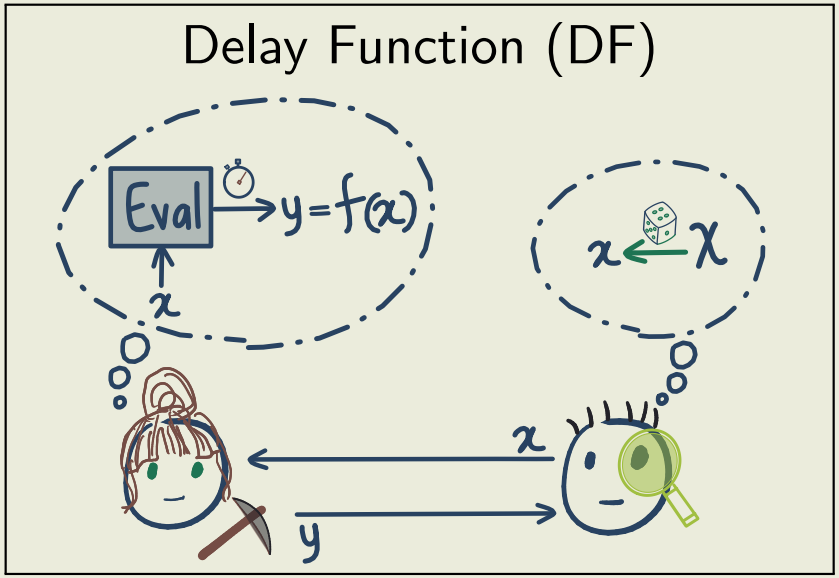
- Simplest example: delay function (DF) $f : \mathcal{X} \to \mathcal{Y}$
  - i) Can be evaluated in time T; but
  - ii) Cannot be evaluated in time $\ll T$ even given parallelism

Alternatively: primitives secure only for a predetermined wall-time T

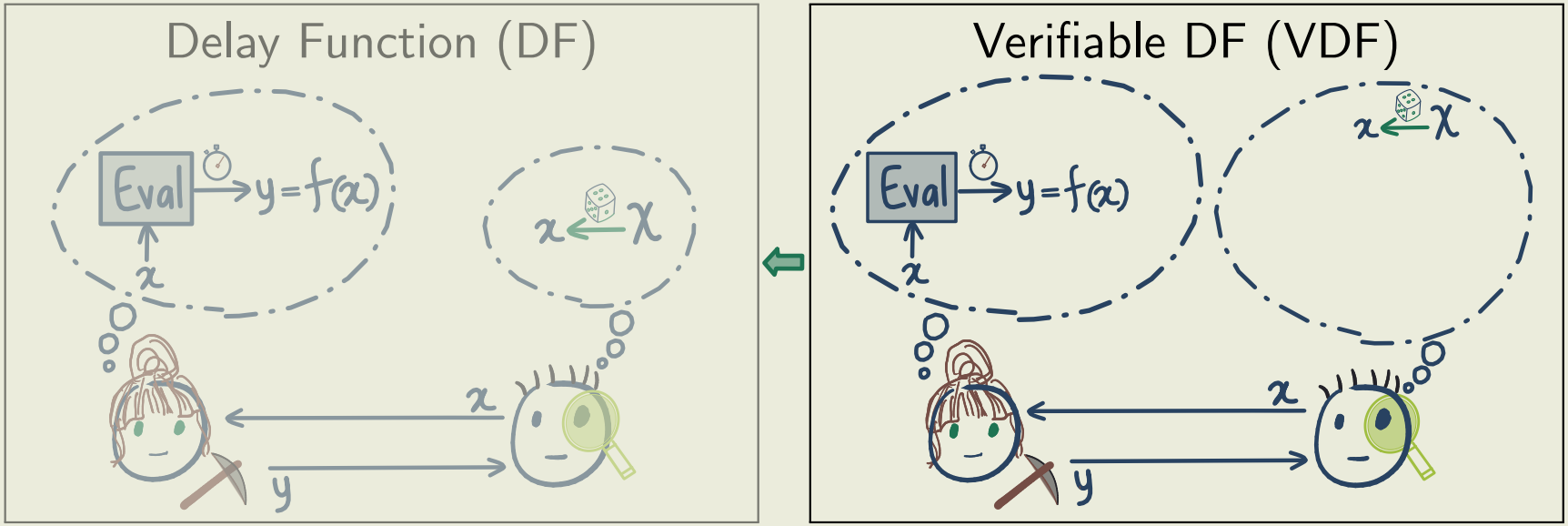
What is a Timed Primitive?..



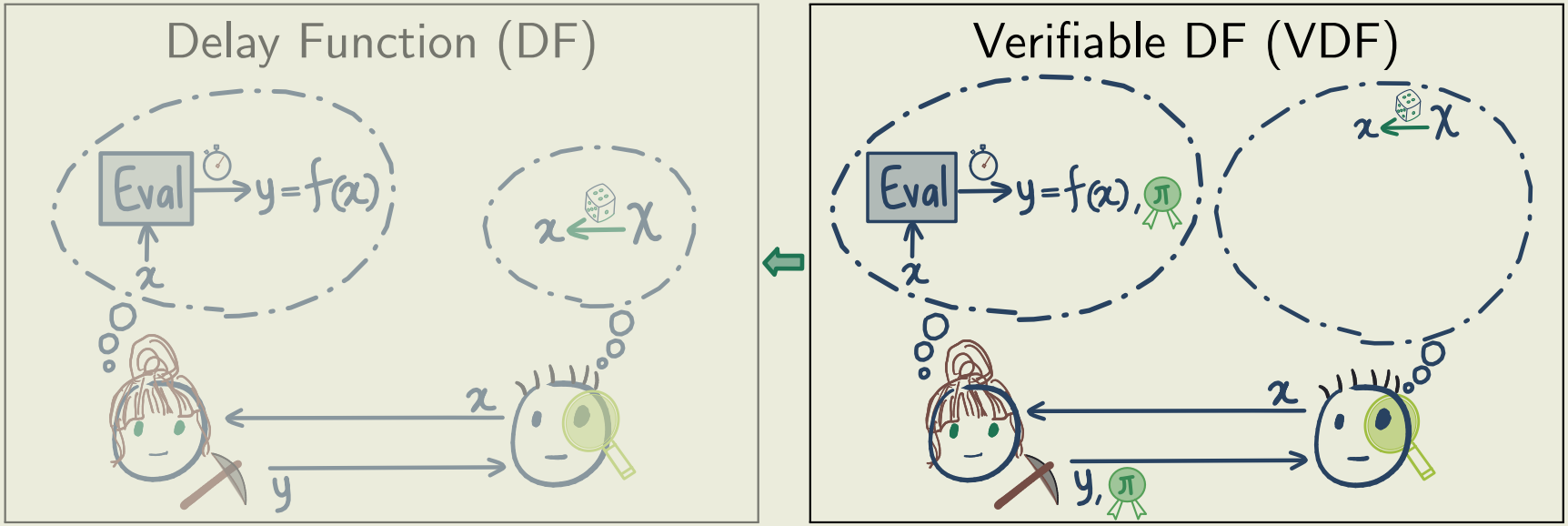
What is a Timed Primitive?..



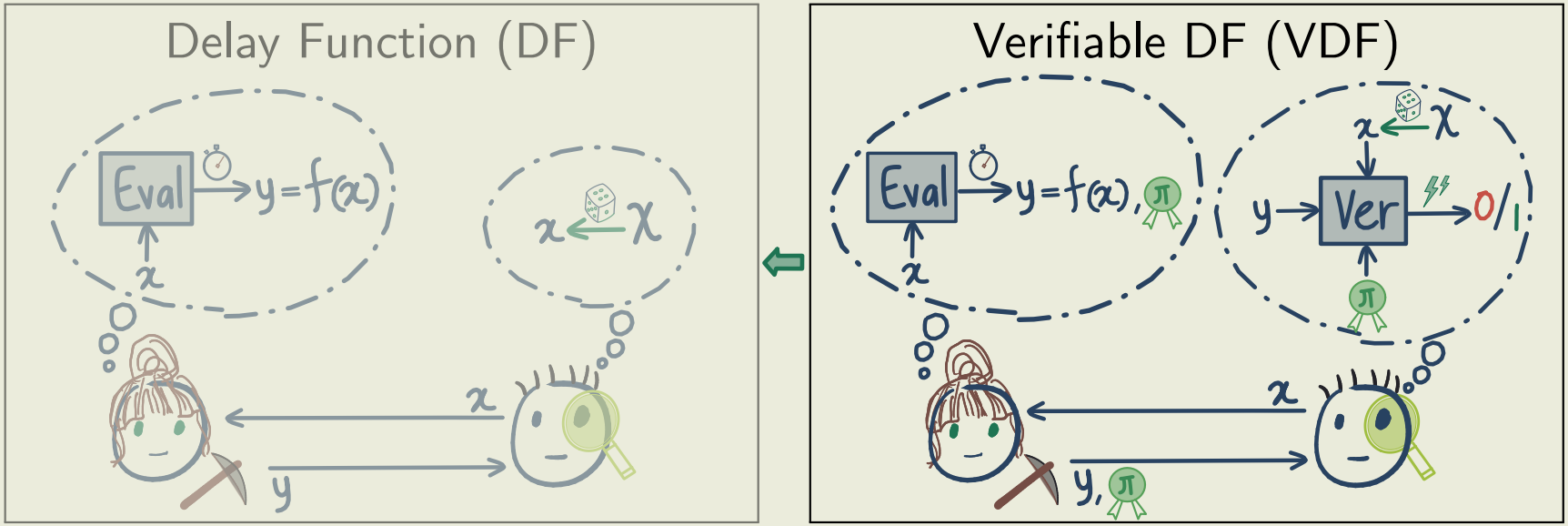
What is a Timed Primitive?..



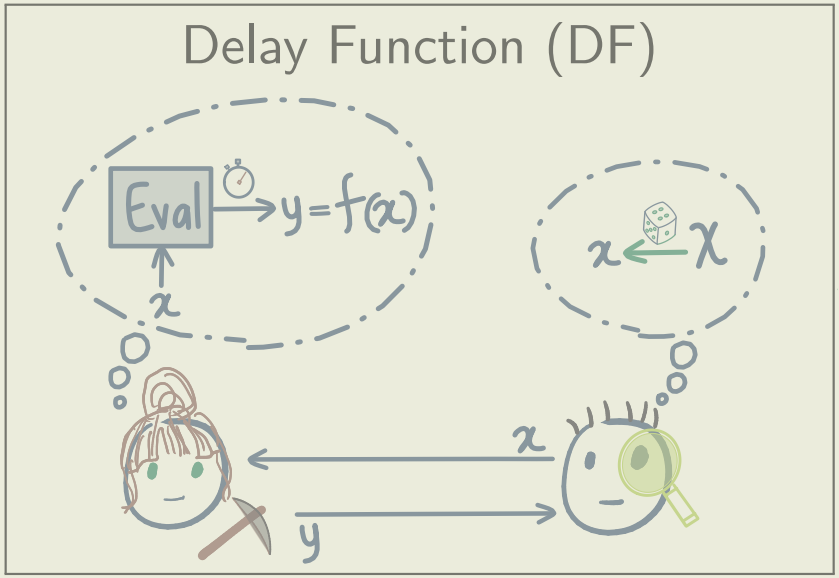
What is a Timed Primitive?..

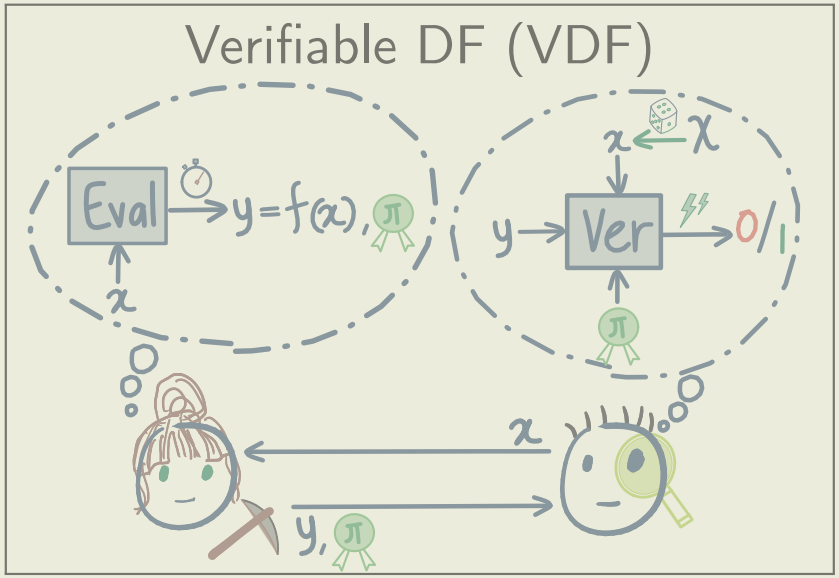


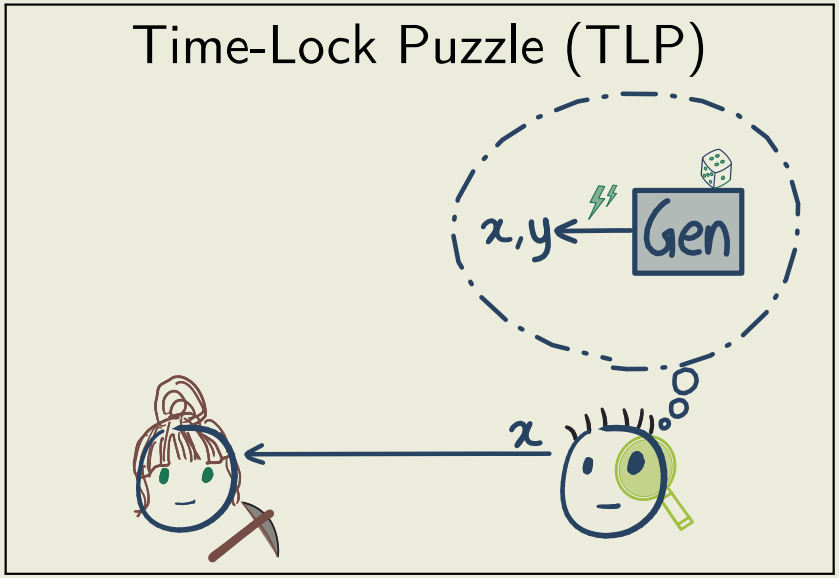
What is a Timed Primitive?..



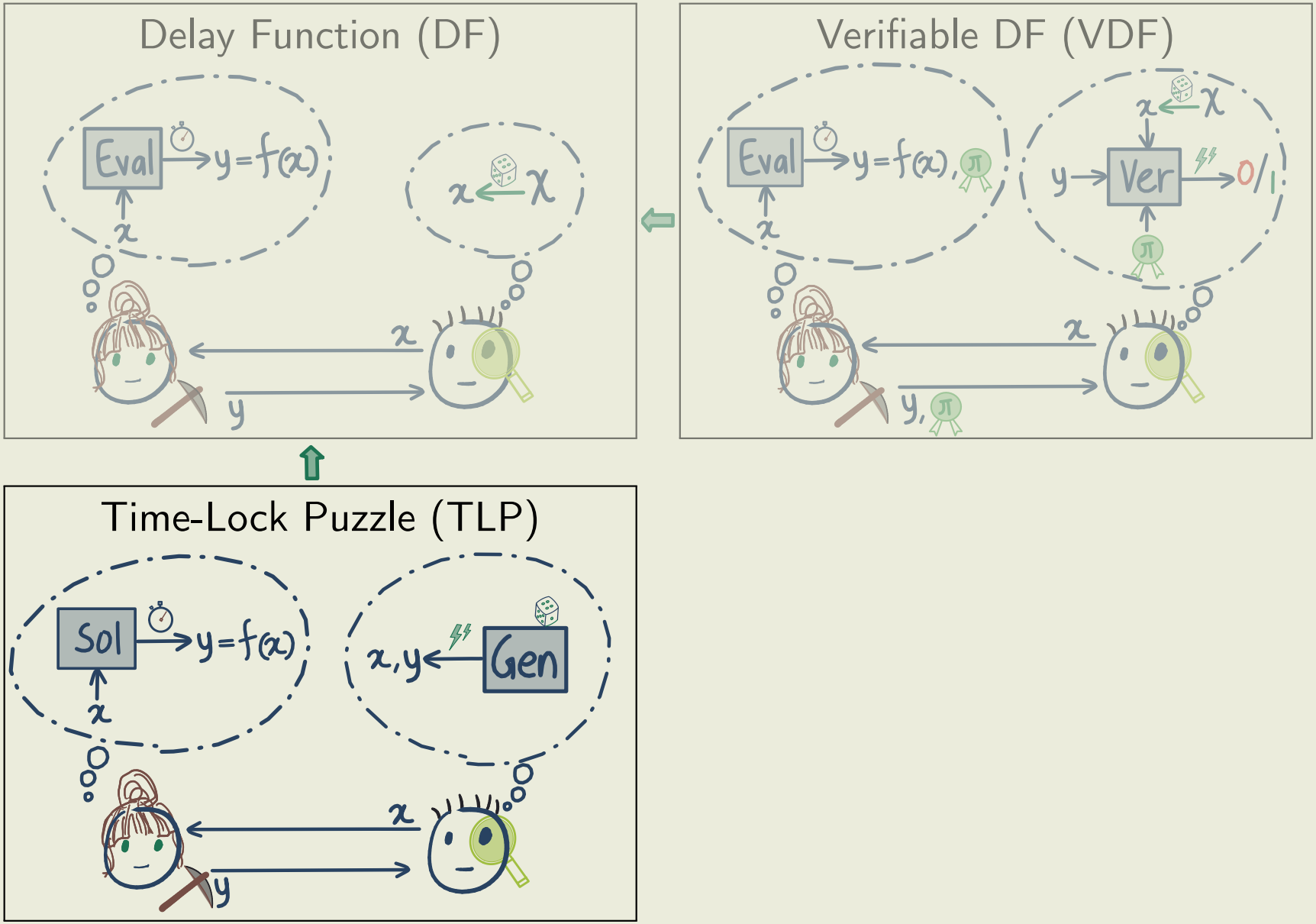
What is a Timed Primitive?..



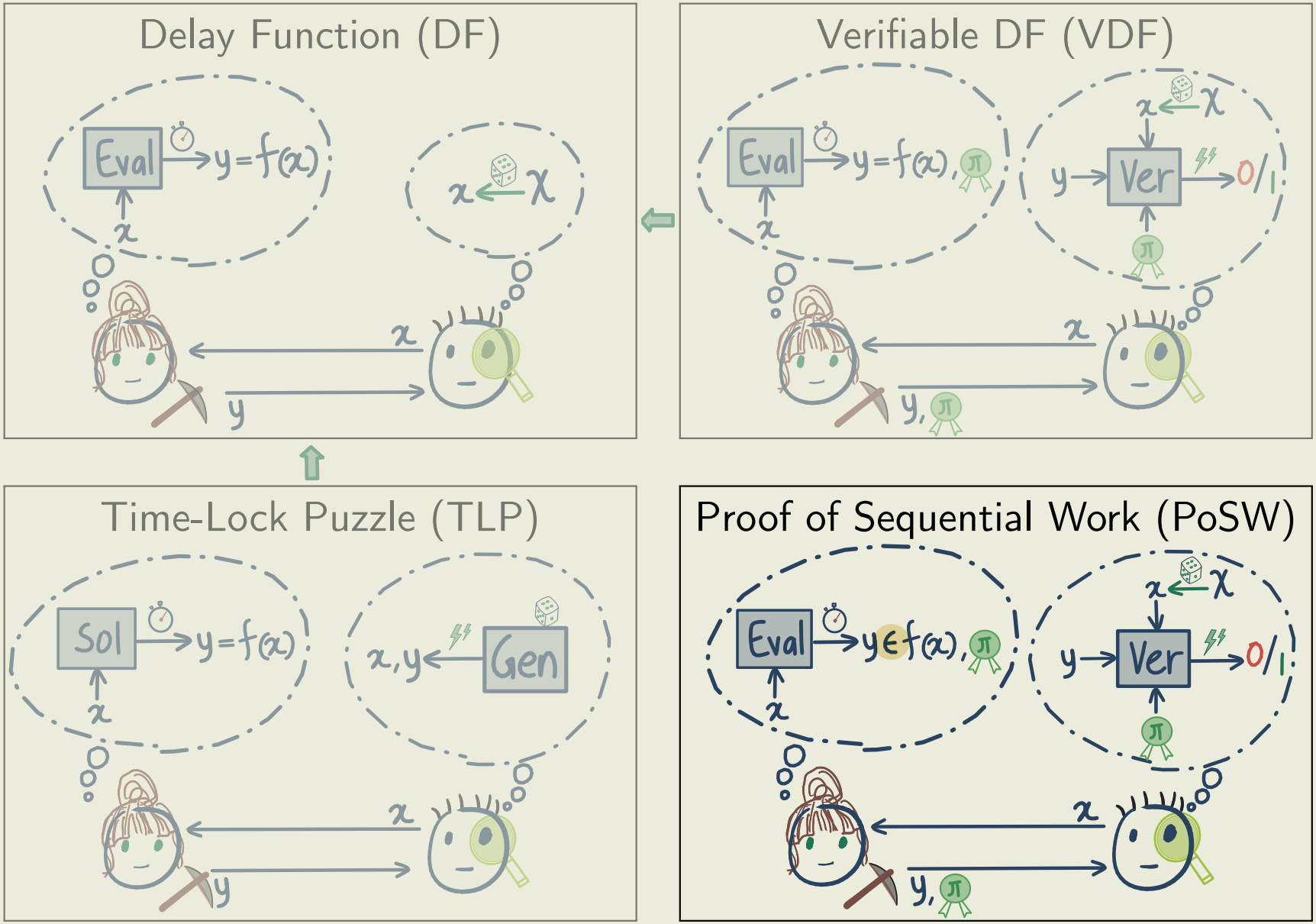




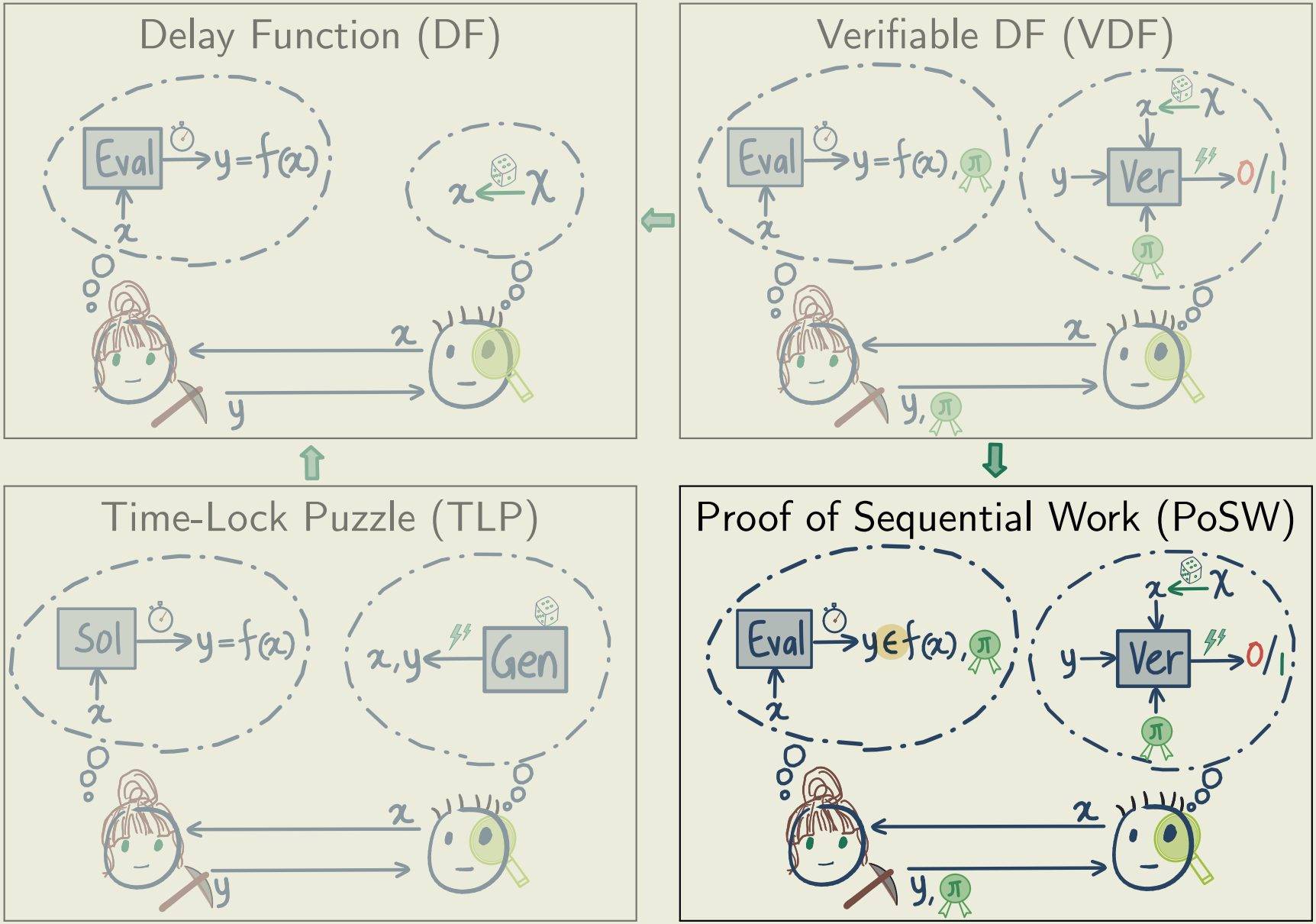
What is a Timed Primitive?..



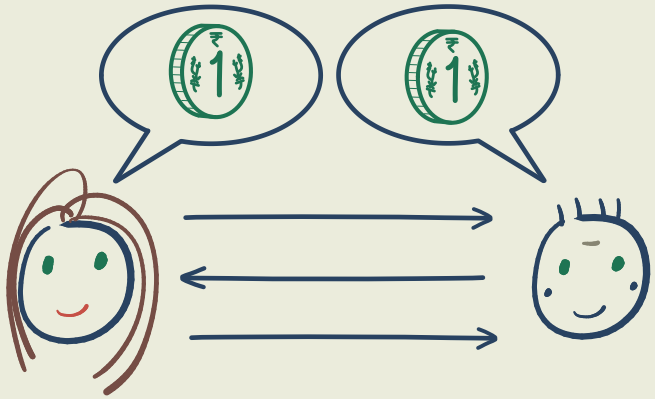
What is a Timed Primitive?..



What is a Timed Primitive?..

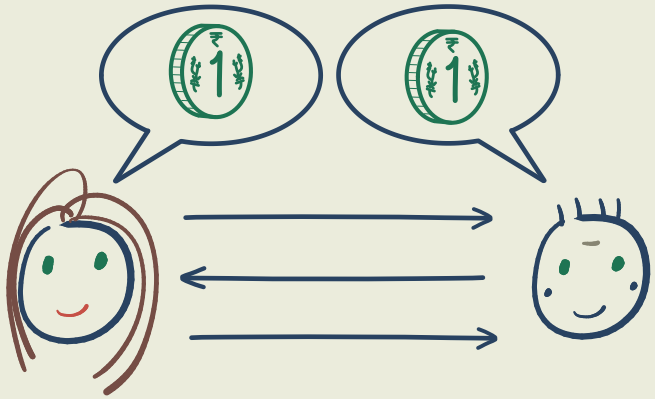


Why are Timed Primitives Useful?



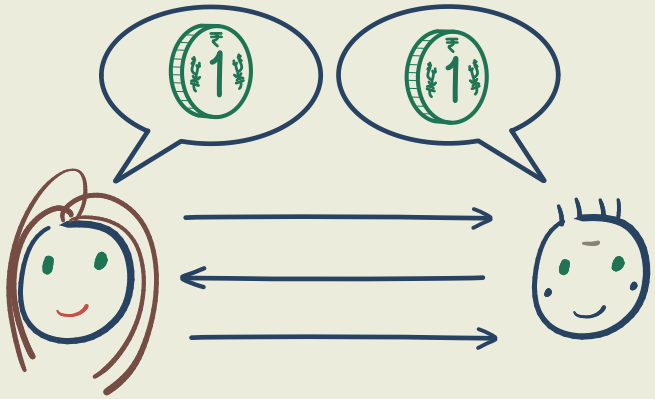
1. Fairness in multi-party protocols, e.g., coin tossing

Why are Timed Primitives Useful?



1. Fairness in multi-party protocols, e.g., coin tossing
   - ! Impossible in the plain model[Cle86]
   - ✓ Possible given a TLP![GS98, FKPS21]

Why are Timed Primitives Useful?



1. Fairness in multi-party protocols, e.g., coin tossing
   - ! Impossible in the plain model[Cle86]
   - ✓ Possible given a TLP![GS98, FKPS21]
2. Randomness beacons[LW17]
3. Benchmarking hardware[CLSY93]
4. Deniability, via short-lived signatures[ABC22]

...

Plan for Today's Talk

Timed Cryptography



Current Landscape

Breaking Random-Oracle Based VDF

What Can Timed Primitives Be Constructed From?..

1. Integer factorisation 

What Can Timed Primitives Be Constructed From?..

1. Integer factorisation 

    DF[RSW96]: for $N := pq$,

$$f_{N,T}(x) := x^{2^T} \bmod N$$

What Can Timed Primitives Be Constructed From?..

1. Integer factorisation 



DF[RSW96]: for $N := pq$,

$$f_{N,T}(x) := x^{2^T} \bmod N$$

- Honest evaluation using T squarings modulo N:

$$x \to x^2 \to x^{2^2} \to \cdots \to x^{2^{T-2}} \to x^{2^{T-1}} \to x^{2^T} \bmod N$$



RSW assumption: without N's factors, T squaring are necessary to compute $f_{N,T}$

What Can Timed Primitives Be Constructed From?..

1. Integer factorisation 



DF[RSW96]: for $N := pq$,

$$f_{N,T}(x) := x^{2^T} \bmod N$$

- Honest evaluation using T squarings modulo N:

$$x \longrightarrow x^2 \longrightarrow x^{2^2} \longrightarrow \cdots \longrightarrow x^{2^{T-2}} \longrightarrow x^{2^{T-1}} \longrightarrow x^{2^T} \bmod N$$



RSW assumption: without N's factors, T squaring are necessary to compute $f_{N,T}$



TLP[RSW96]:

- Puzzle is $x \leftarrow \mathbb{Z}_N^\times$ and its solution is $f_{N,T}(x)$

What Can Timed Primitives Be Constructed From?..

1. Integer factorisation 



DF[RSW96]: for $N := pq$,

$$f_{N,T}(x) := x^{2^T} \bmod N$$

- Honest evaluation using T squarings modulo N:

$$x \to x^2 \to x^{2^2} \to \cdots \to x^{2^{T-2}} \to x^{2^{T-1}} \to x^{2^T} \bmod N$$



RSW assumption: without N's factors, T squaring are necessary to compute $f_{N,T}$



TLP[RSW96]:

- Puzzle is $x \leftarrow \mathbb{Z}_N^\times$ and its solution is $f_{N,T}(x)$
- Puzzle can be "pre-solved" given p and q in time $\approx \log(T)$: compute $e := 2^T \bmod (p-1)(q-1)$ and output $x^e \bmod N$

What Can Timed Primitives Be Constructed From?..

1. Integer factorisation 


DF[RSW96]: for $N := pq$,

$$f_{N,T}(x) := x^{2^T} \bmod N$$

- Honest evaluation using T squarings modulo N:

$$x \to x^2 \to x^{2^2} \to \cdots \to x^{2^{T-2}} \to x^{2^{T-1}} \to x^{2^T} \bmod N$$


RSW assumption: without N's factors, T squaring are necessary to compute $f_{N,T}$

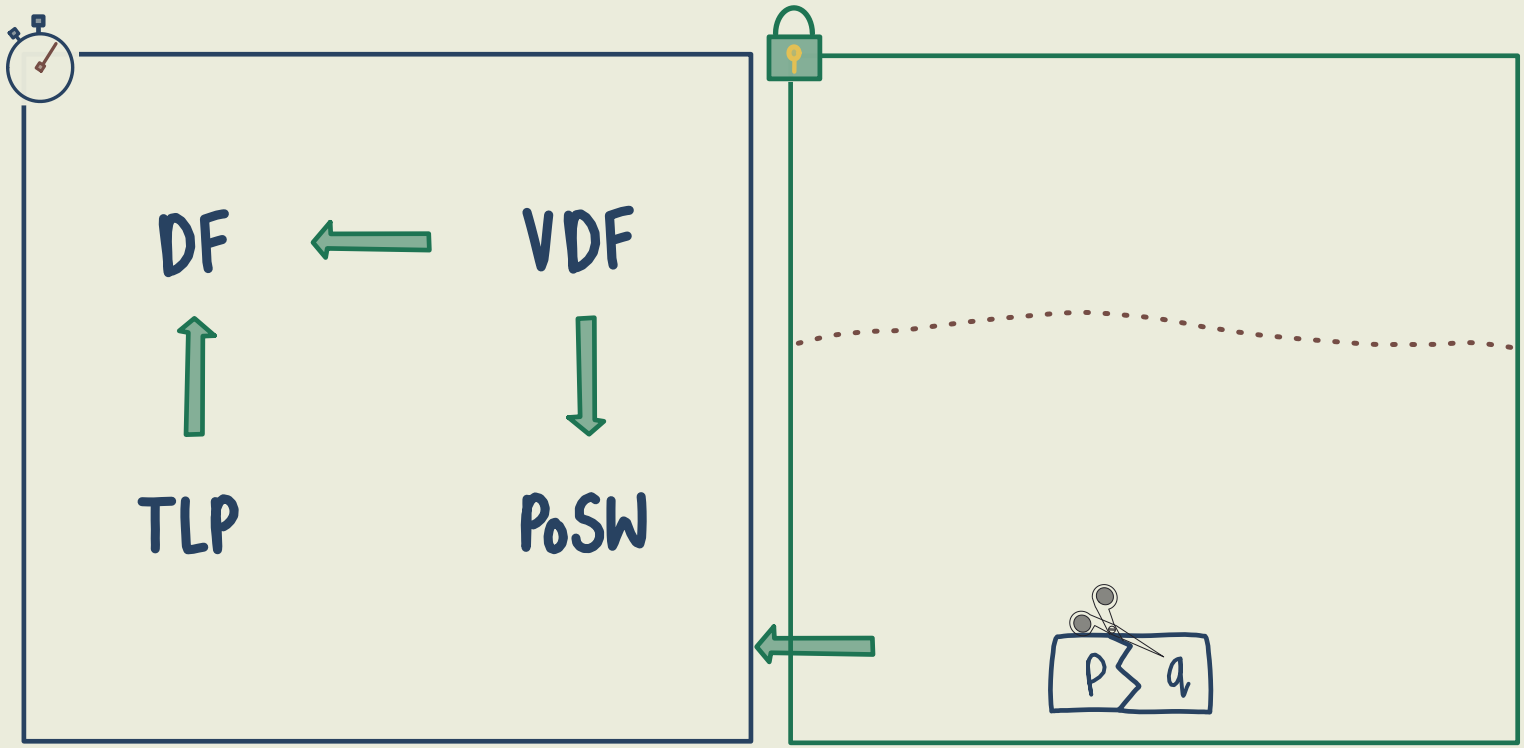

TLP[RSW96]:

- Puzzle is $x \leftarrow \mathbb{Z}_N^\times$ and its solution is $f_{N,T}(x)$
- Puzzle can be "pre-solved" given p and q in time $\approx \log(T)$: compute $e := 2^T \bmod (p-1)(q-1)$ and output $x^e \bmod N$


VDF[Pie19]: to get public verifiability for $f_{N,T}$,

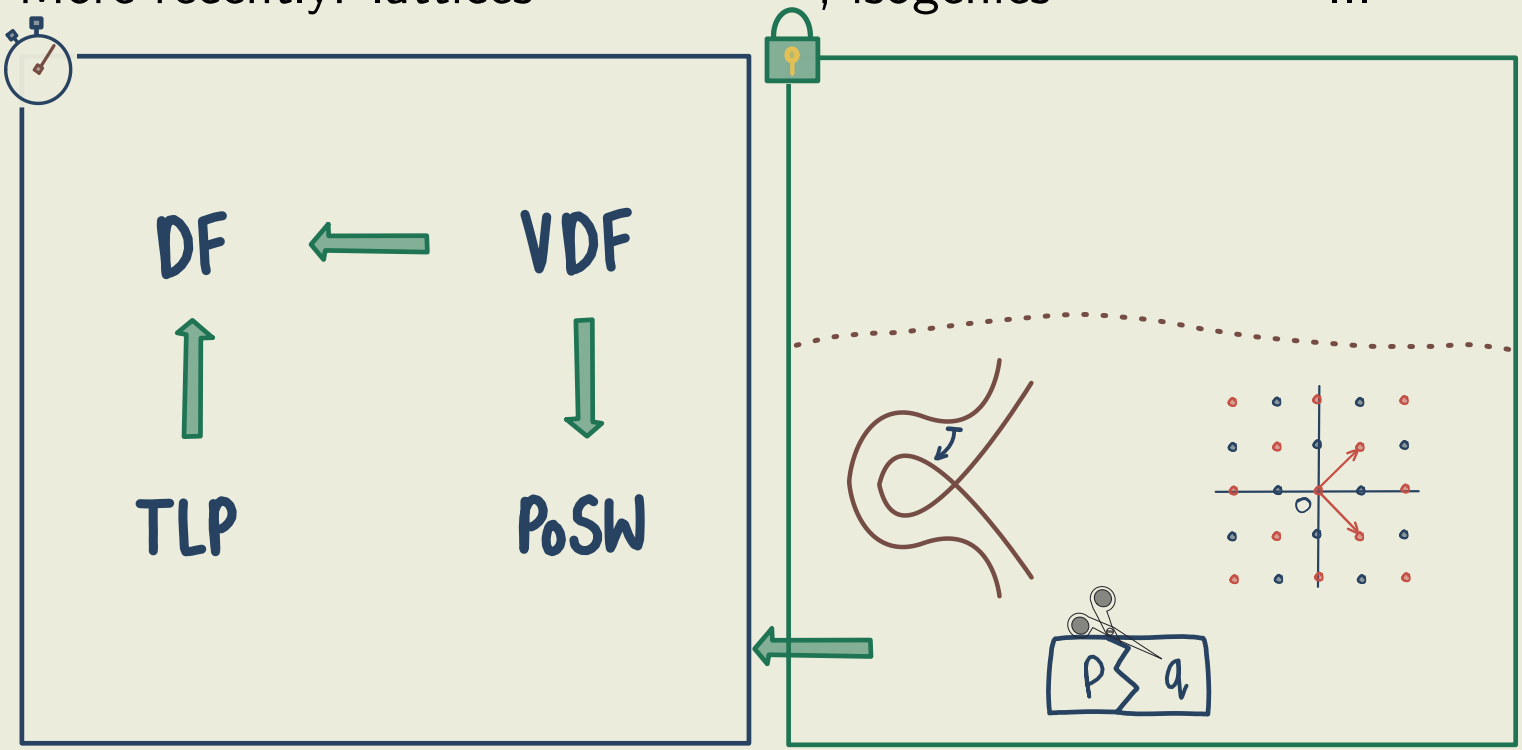
i) Design public-coin interactive proof Π for claim "$y = f_{N,T}(x)$"
ii) Make non-interactive using Fiat-Shamir, e.g., using random oracle

What Can Timed Primitives Be Constructed From?..



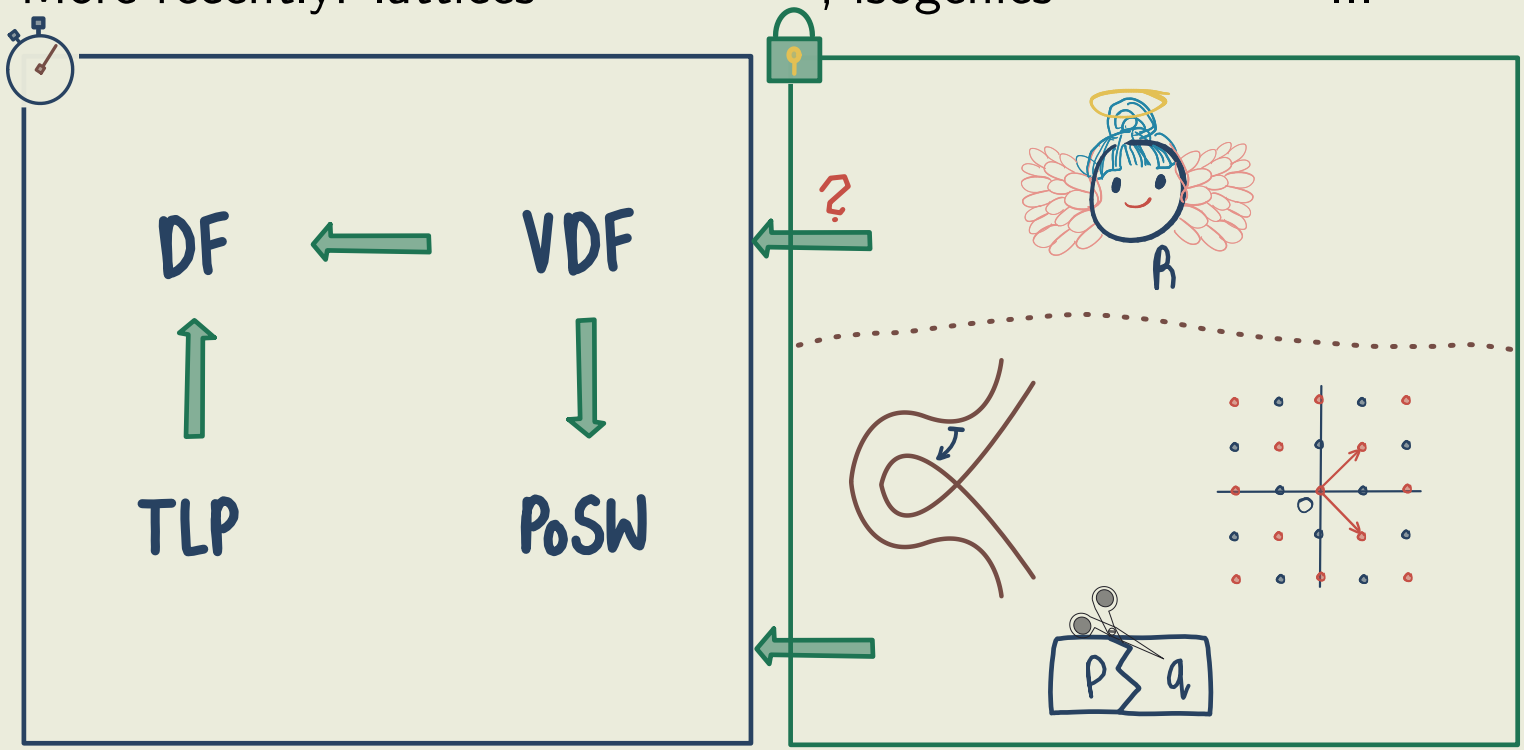
What Can Timed Primitives Be Constructed From?..

2. More recently: lattices [LM19, AMZ24], isogenies [BD21, ABP26]...



What Can Timed Primitives Be Constructed From?..

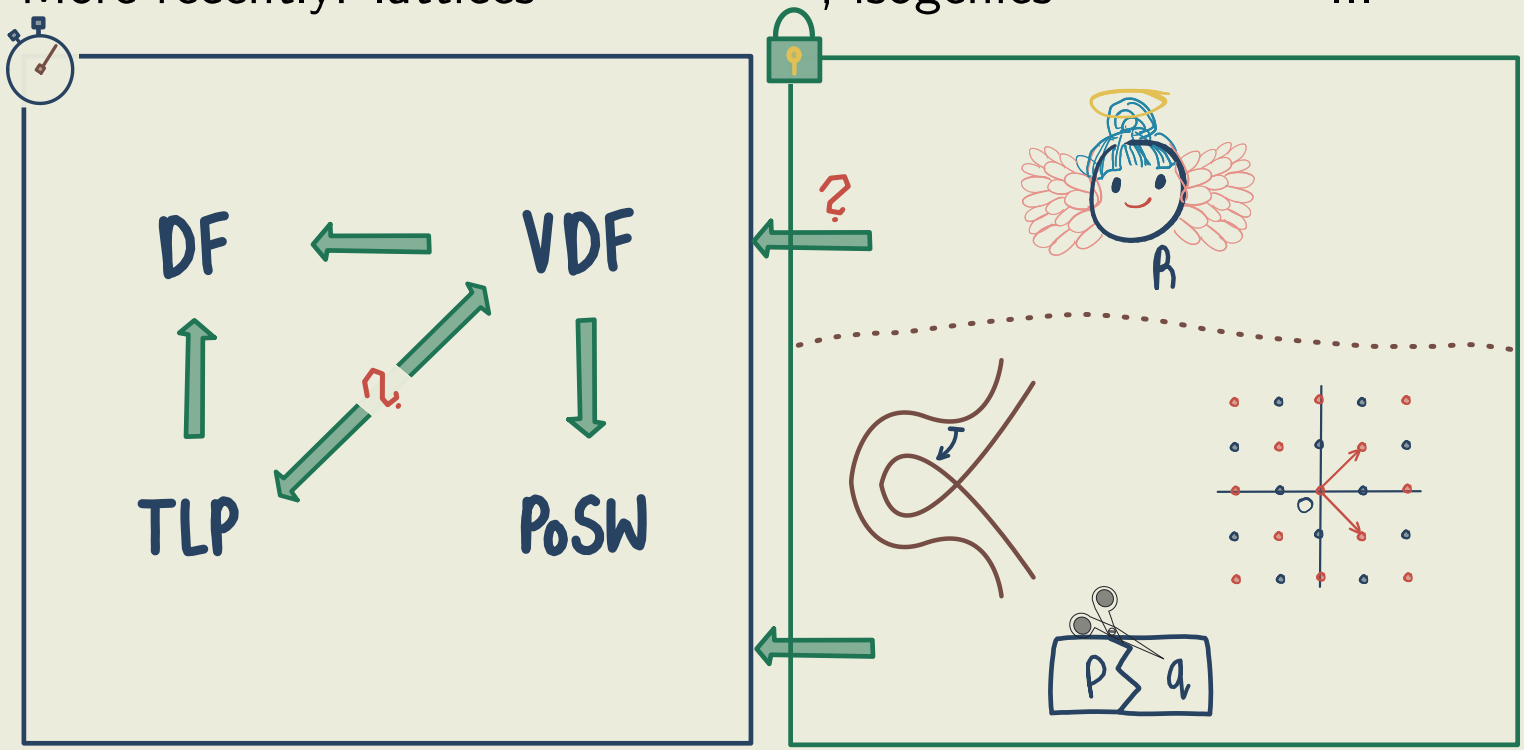
2. More recently: lattices [LM19, AMZ24], isogenies [BD21, ABP26]...



3. What about unstructured primitives, say (only) a random oracle?

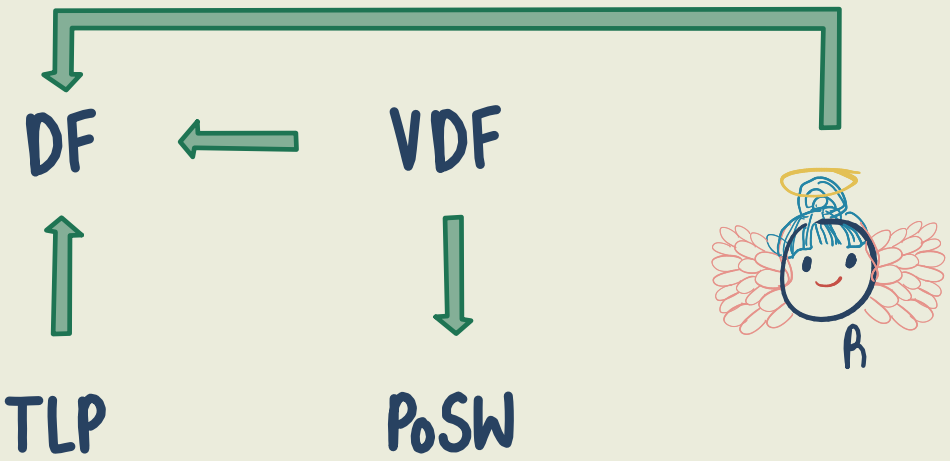
What Can Timed Primitives Be Constructed From?..

2. More recently: lattices [LM19, AMZ24], isogenies [BD21, ABP26]...



3. What about unstructured primitives, say (only) a random oracle?
4. What about *other* timed primitives?
   i) Can we construct VDF from TLP?
   ii) And vice versa?

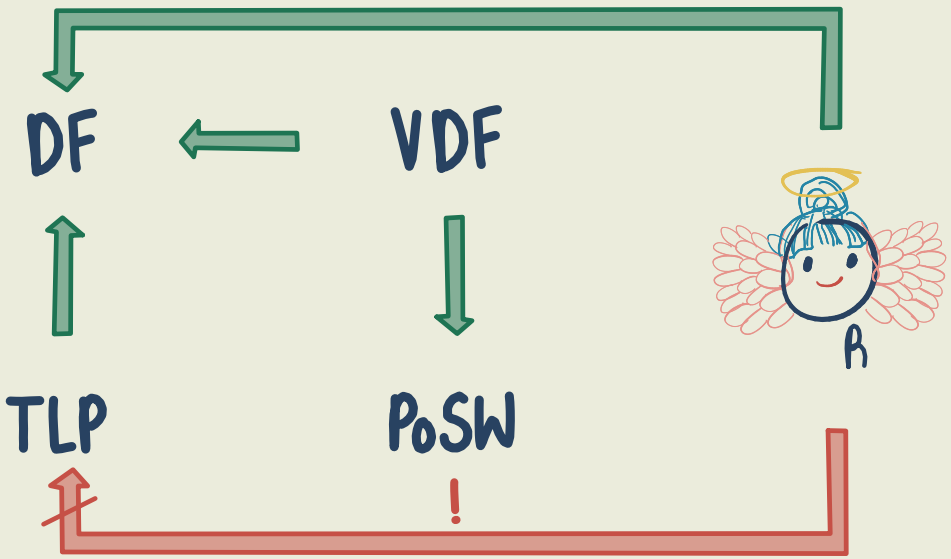
Timed Primitives from Random Oracle (RO)



- DF[FM97] is possible: compose a (length-preserving) RO R

$$f_T^R(x) := R^T(x)$$

Timed Primitives from Random Oracle (RO)

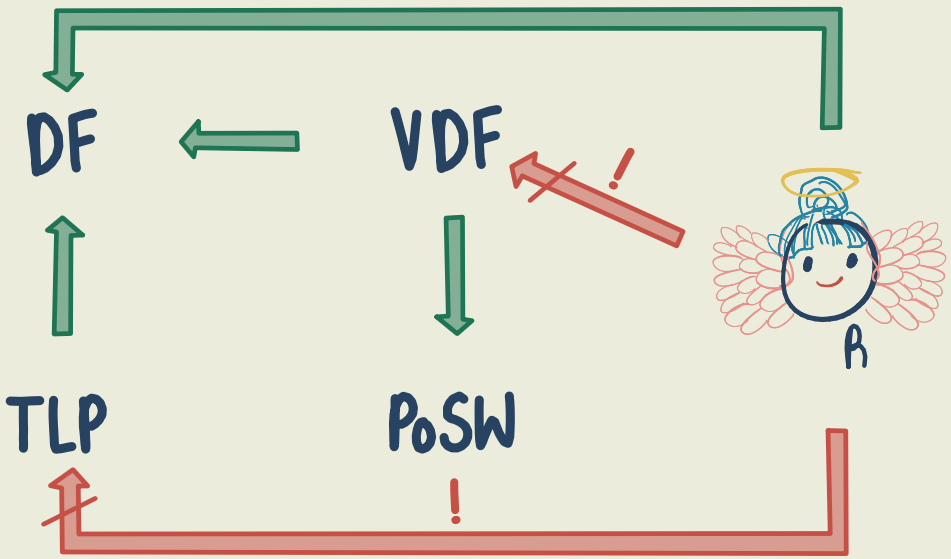


- DF[FM97] is possible: compose a (length-preserving) RO R

$$f_T^R(x) := R^T(x)$$

- TLP is not possible[MMV11]: *any TLP in which* Gen *makes at most* q *queries to* R *can be solved in* $\approx q$ *time (i.e., rounds of queries)*

Timed Primitives from Random Oracle (RO)

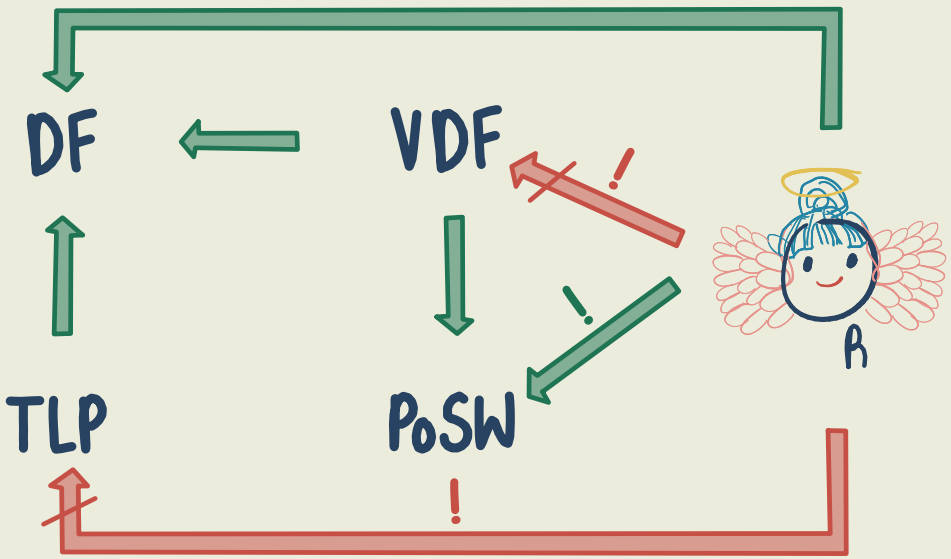


- DF[FM97] is possible: compose a (length-preserving) RO R

$$f_T^R(x) := R^T(x)$$

- TLP is not possible[MMV11]: *any TLP in which* Gen *makes at most* q *queries to* R *can be solved in* $\approx q$ *time (i.e., rounds of queries)*
- VDF too is not possible[MSW20, GRY25, AAK+26]: coming up

Timed Primitives from Random Oracle (RO)

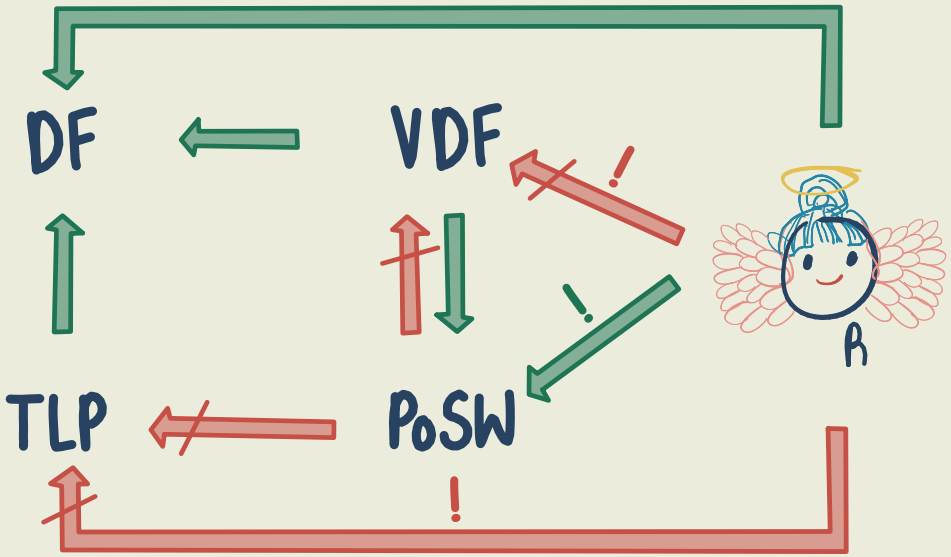


- DF[FM97] is possible: compose a (length-preserving) RO R

$$f_T^R(x) := R^T(x)$$

- TLP is not possible[MMV11]: *any TLP in which* Gen *makes at most* q *queries to* R *can be solved in* $\approx q$ *time (i.e., rounds of queries)*
- VDF too is not possible[MSW20, GRY25, AAK+26]: coming up
- PoSW is possible![MMV13, CP18]

Timed Primitives from Random Oracle (RO)

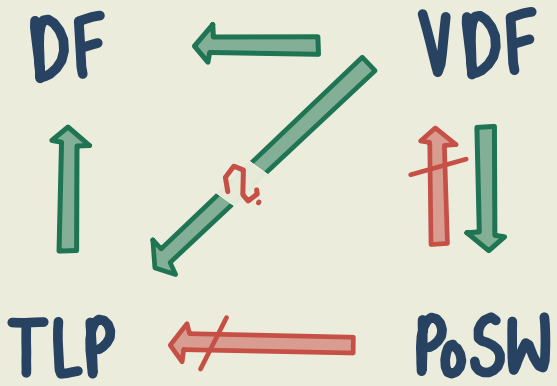


- DF[FM97] is possible: compose a (length-preserving) RO R

$$f_T^R(x) := R^T(x)$$

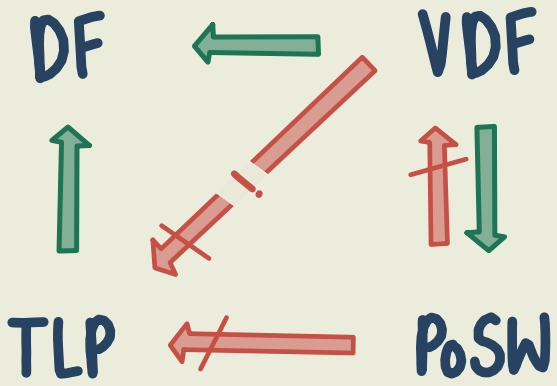
- TLP is not possible[MMV11]: *any TLP in which* Gen *makes at most* q *queries to* R *can be solved in* $\approx q$ *time (i.e., rounds of queries)*
- VDF too is not possible[MSW20, GRY25, AAK+26]: coming up
- PoSW is possible![MMV13, CP18]

Timed Primitives from Other Timed Primitives..



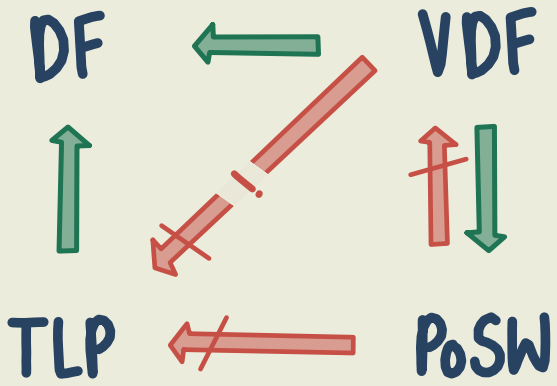
1. VDF $\Rightarrow$ TLP?

Timed Primitives from Other Timed Primitives..



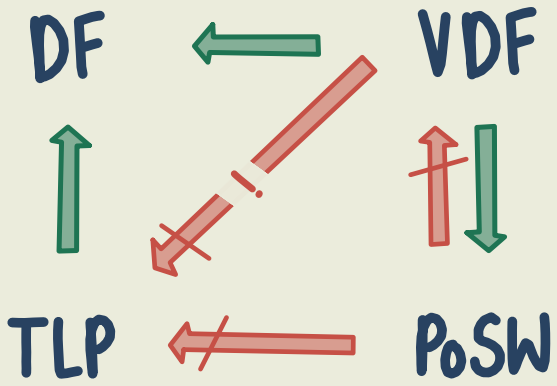
1. VDF $\Rightarrow$ TLP? Not possible: *there exists a world* $W_1 := (R, V_R)$ *in which a VDF exists but TLPs don't.*[AAA+26]

Timed Primitives from Other Timed Primitives..



1. VDF $\Rightarrow$ TLP? Not possible: *there exists a world* $W_1 := (R, V_R)$ *in which a VDF exists but TLPs don't.*[AAA+26]
   - R is a RO and V_R is an auxiliary "verifier" oracle for which $V_R(x, y, T) = 1$ holds iff $R^T(x) = y$

Timed Primitives from Other Timed Primitives..



1. VDF $\Rightarrow$ TLP? Not possible: *there exists a world* $W_1 := (R, V_R)$ *in which a VDF exists but TLPs don't.*[AAA+26]

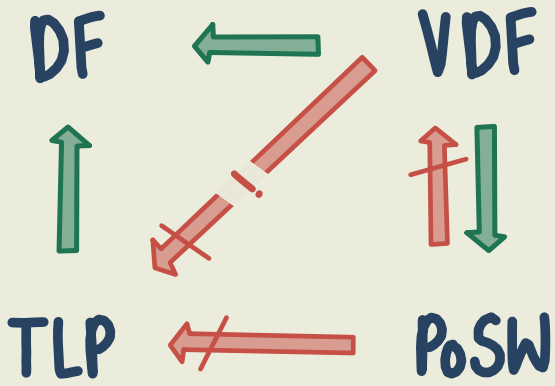
   - R is a RO and V_R is an auxiliary "verifier" oracle for which $V_R(x, y, T) = 1$ holds iff $R^T(x) = y$

   + Why VDF trivially exists in W_1?

     i) Delay function can be defined as $R^T(x)$

     ii) Verifiability is free given V_R

Timed Primitives from Other Timed Primitives..



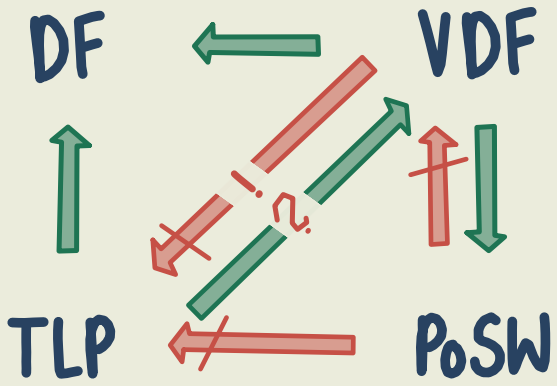
1. VDF $\Rightarrow$ TLP? Not possible: *there exists a world $W_1 := (R, V_R)$ in which a VDF exists but TLPs don't.*[AAA+26]

   - R is a RO and V_R is an auxiliary "verifier" oracle for which $V_R(x, y, T) = 1$ holds iff $R^T(x) = y$

   + Why VDF trivially exists in W_1?
     i) Delay function can be defined as $R^T(x)$
     ii) Verifiability is free given V_R

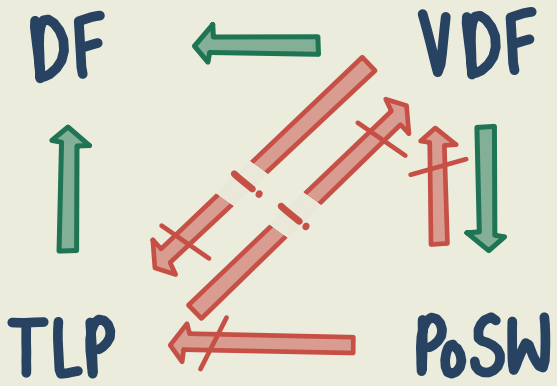
   ! Why TLP cannot exist in W_1?
     i) The use of V_R in the TLP construction can be "simulated"
     ii) Hence TLP can be constructed only from R (not possible[MMV11])

Timed Primitives from Other Timed Primitives..



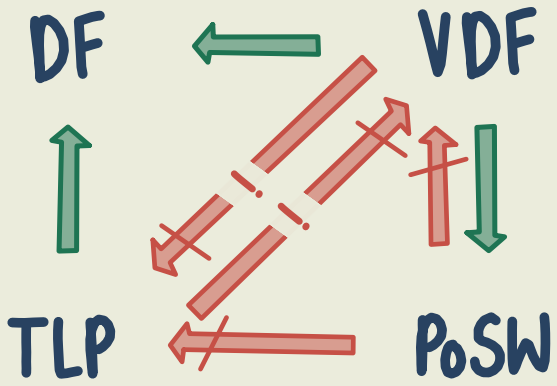
2. VDF $\Leftarrow$ TLP?

Timed Primitives from Other Timed Primitives..



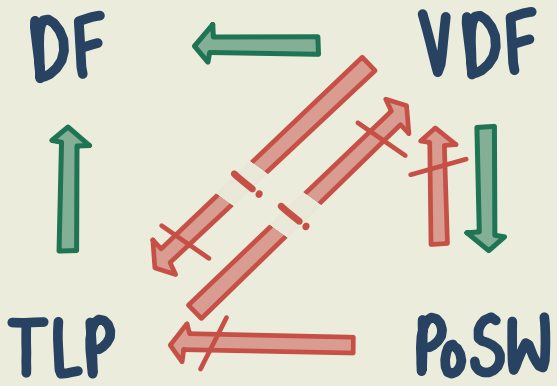
2. VDF $\Leftarrow$ TLP? Not possible: *there exists a world* $W_2 := (P, P^{-1}, S_P)$ *in which a TLP exists but VDFs don't.*[AAA+26]

Timed Primitives from Other Timed Primitives..



2. VDF $\Leftarrow$ TLP? Not possible: *there exists a world* $W_2 := (P, P^{-1}, S_P)$ *in which a TLP exists but VDFs don't.*[AAA+26]
   - P is a random permutation, P^{-1} is its inverse and S_P is an auxiliary "shortcut" oracle such that that $S_P(x, t) := P^t(x)$

Timed Primitives from Other Timed Primitives..



2. VDF $\Leftarrow$ TLP? Not possible: *there exists a world* $W_2 := (P, P^{-1}, S_P)$ *in which a TLP exists but VDFs don't.*[AAA+26]

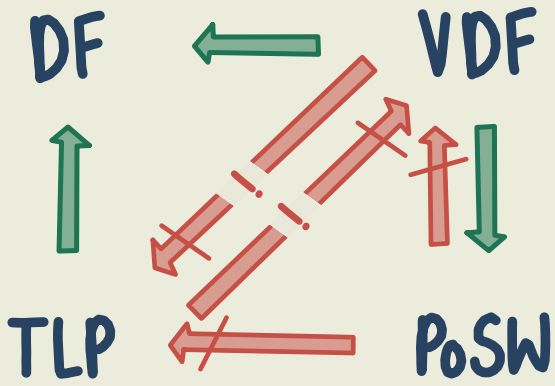
   - P is a random permutation, P^{-1} is its inverse and S_P is an auxiliary "shortcut" oracle such that that $S_P(x, t) := P^t(x)$

   + Why TLP trivially exists in W_2?

     i) Generating a puzzle with solution is free given S_P: sample solution x and set $y := S_P(x, T)$ as the puzzle

     ii) Solve puzzle using T sequential calls to P^{-1}

Timed Primitives from Other Timed Primitives..



2. VDF $\Leftarrow$ TLP? Not possible: *there exists a world* $W_2 := (P, P^{-1}, S_P)$ *in which a TLP exists but VDFs don't.*[AAA+26]

   - P is a random permutation, P^{-1} is its inverse and S_P is an auxiliary "shortcut" oracle such that that $S_P(x, t) := P^t(x)$

   + Why TLP trivially exists in W_2?
     i) Generating a puzzle with solution is free given S_P: sample solution x and set $y := S_P(x, T)$ as the puzzle
     ii) Solve puzzle using T sequential calls to P^{-1}

   ! Why VDF cannot exist in W_2? Extend the ideas of why RO-based VDFs can be broken[AAA+26]

Plan for Today's Talk

Timed Cryptography

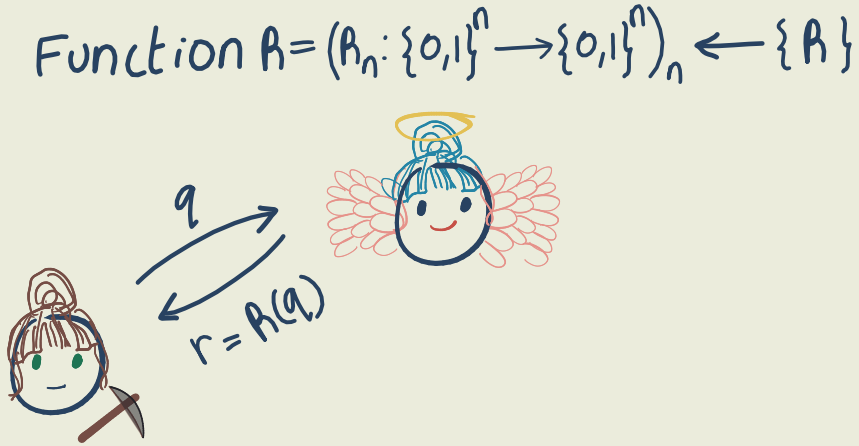


Current Landscape



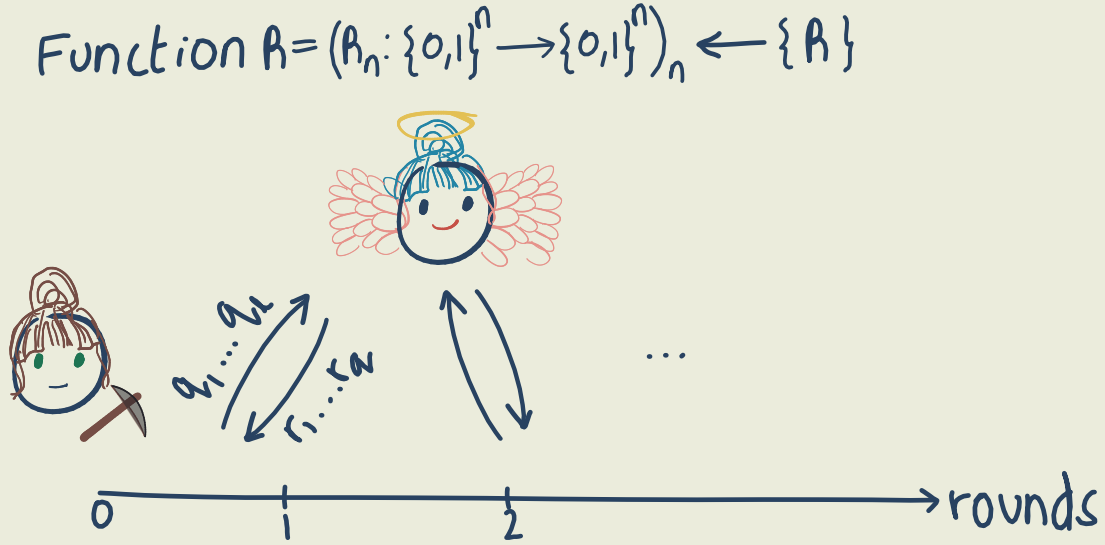
Breaking Random-Oracle Based VDF

(Parallel) Random-Oracle Model



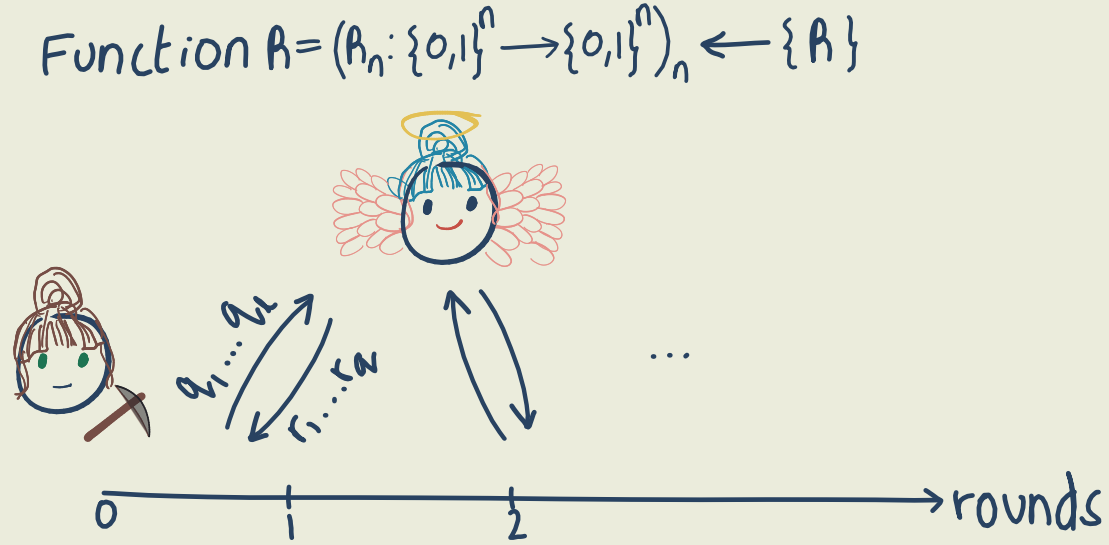
- All algorithms have query access to a RO R

(Parallel) Random-Oracle Model



- All algorithms have query access to a RO R
- An algorithm runs in rounds: in each round it can query R *in parallel*

(Parallel) Random-Oracle Model



- All algorithms have query access to a RO R
- An algorithm runs in rounds: in each round it can query R *in parallel*

₹ Complexity measures for an algorithm:

- ⏱ Round complexity: total number of *rounds* in which it queries R
- Σ Query complexity: total number of *queries* it makes to R

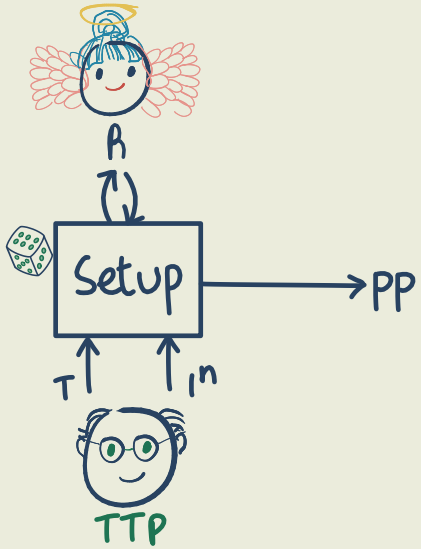
RO-Based VDF

is a triple of oracle algorithms $(\mathsf{Setup}^{\cdot}, \mathsf{Eval}^{\cdot}, \mathsf{Ver}^{\cdot})$ with following syntax:



RO-Based VDF

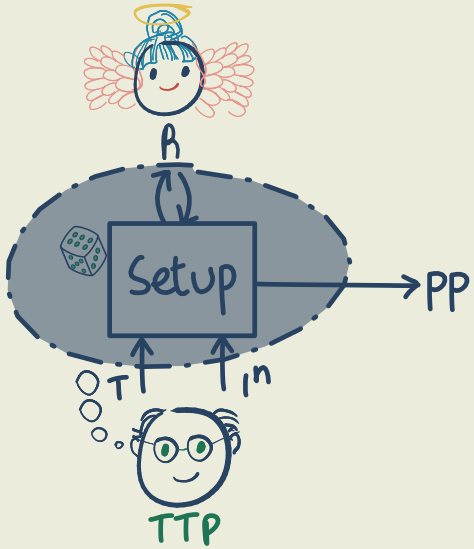
is a triple of oracle algorithms $(\mathsf{Setup}^{\cdot}, \mathsf{Eval}^{\cdot}, \mathsf{Ver}^{\cdot})$ with following syntax:



RO-Based VDF

↗ Public vs. private

is a triple of oracle algorithms $(\mathsf{Setup}, \mathsf{Eval}, \mathsf{Ver})$ with following syntax:

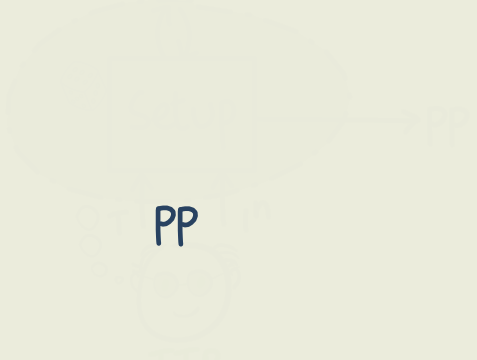


RO-Based VDF

↗ Public vs. private

is a triple of oracle algorithms $(\mathsf{Setup}^{\cdot}, \mathsf{Eval}^{\cdot}, \mathsf{Ver}^{\cdot})$ with following syntax:

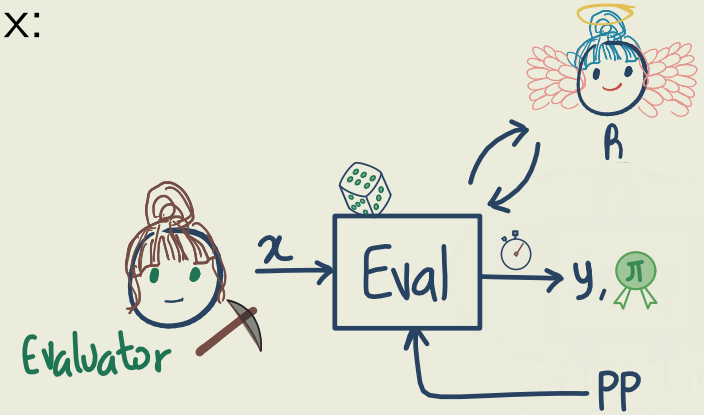




RO-Based VDF

↗ Public vs. private

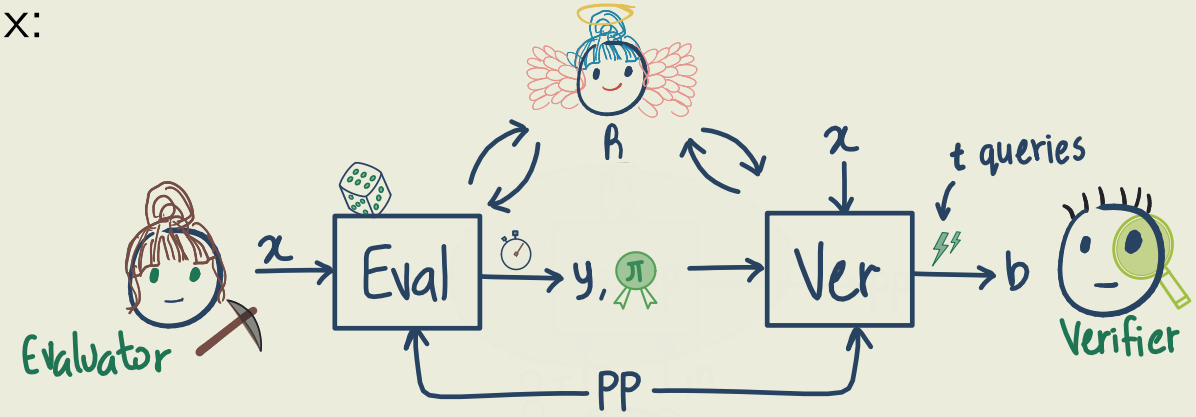
is a triple of oracle algorithms $(\mathsf{Setup}^{\cdot}, \mathsf{Eval}^{\cdot}, \mathsf{Ver}^{\cdot})$ with following syntax:



RO-Based VDF

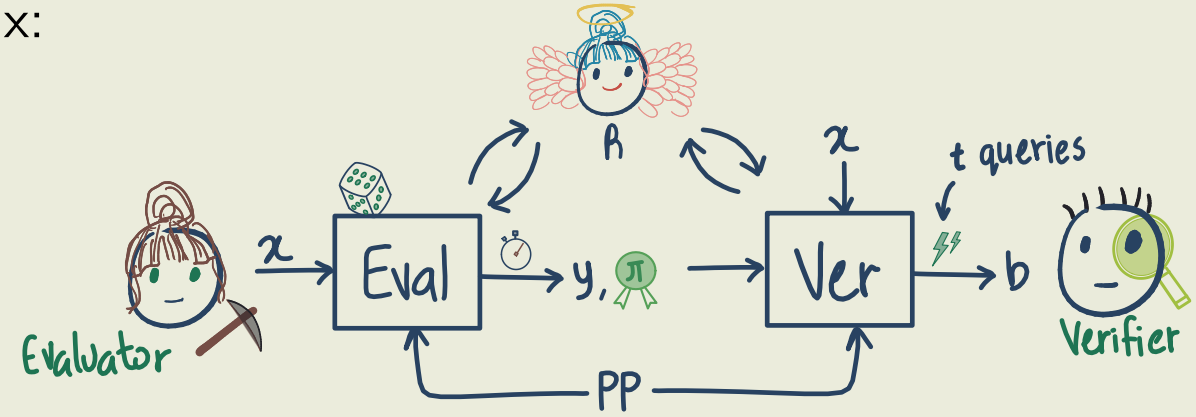
↗ Public vs. private

is a triple of oracle algorithms $(\mathsf{Setup}^{\cdot}, \mathsf{Eval}^{\cdot}, \mathsf{Ver}^{\cdot})$ with following syntax:



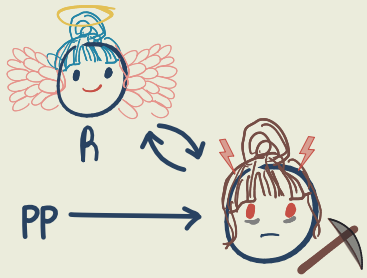
RO-Based VDF

is a triple of oracle algorithms $(\mathsf{Setup}^{\cdot}, \mathsf{Eval}^{\cdot}, \mathsf{Ver}^{\cdot})$ with following syntax:



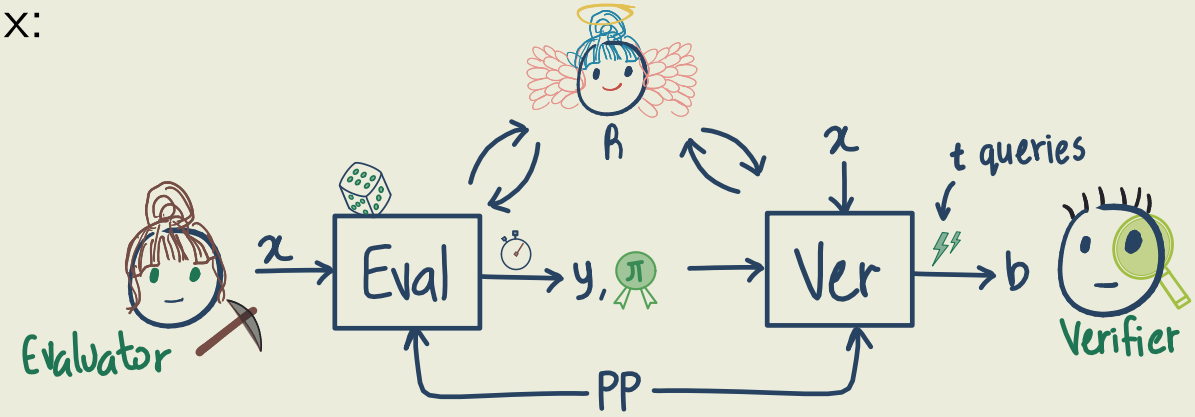
Security requirements:

1. Sequentiality: no $\ll T$-round oracle adversary, given $x \leftarrow \mathcal{X}$, can produce $y := f(x)$, even with preprocessing



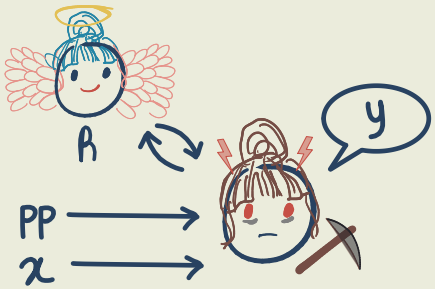
RO-Based VDF

is a triple of oracle algorithms $(\mathsf{Setup}^{\cdot}, \mathsf{Eval}^{\cdot}, \mathsf{Ver}^{\cdot})$ with following syntax:



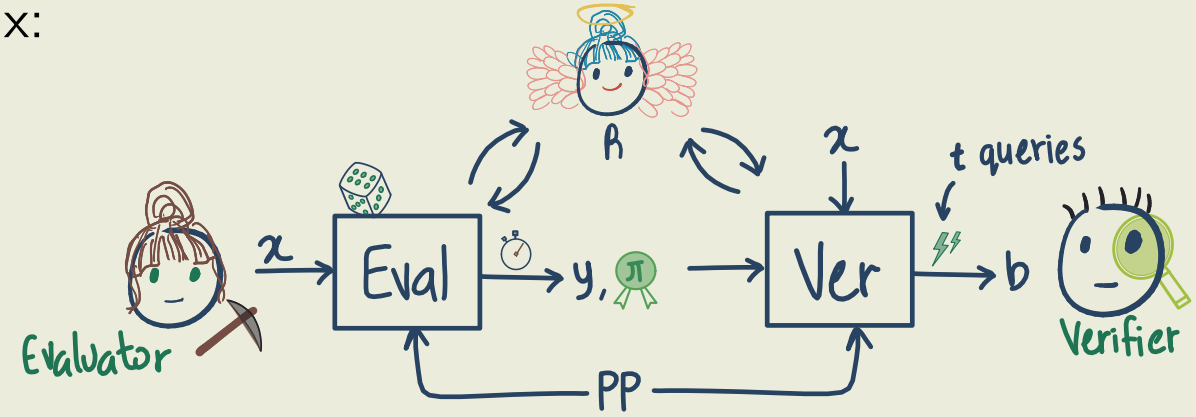
Security requirements:

1. Sequentiality: no $\ll T$-round oracle adversary, given $x \leftarrow \mathcal{X}$, can produce $y := f(x)$, even with preprocessing



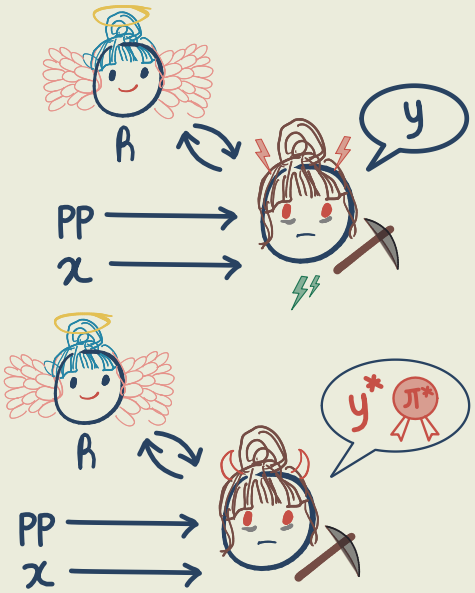
RO-Based VDF

is a triple of oracle algorithms $(\mathsf{Setup}^{\cdot}, \mathsf{Eval}^{\cdot}, \mathsf{Ver}^{\cdot})$ with following syntax:



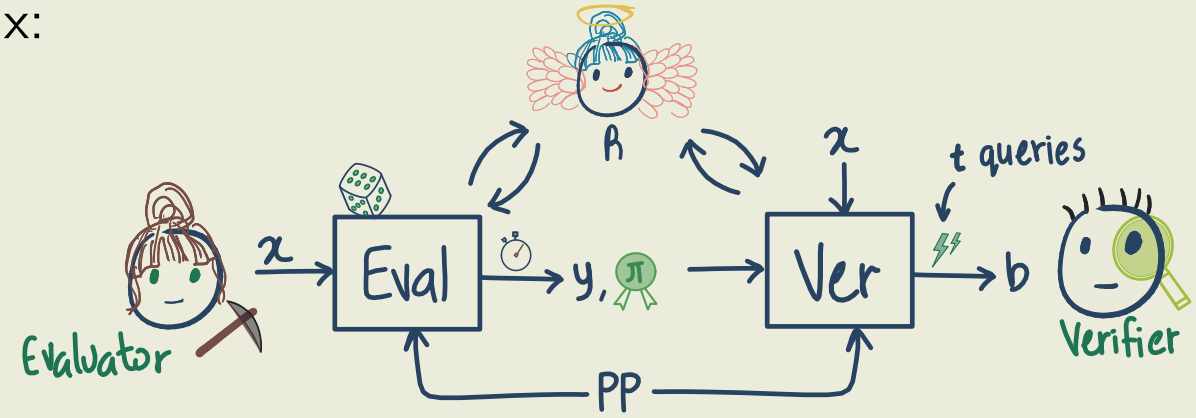
Security requirements:

1. Sequentiality: no $\ll T$-round oracle adversary, given $x \leftarrow \mathcal{X}$, can produce $y := f(x)$, even with preprocessing
2. Perfect uniqueness: no adversary, given $x \leftarrow \mathcal{X}$, can produce (y^*, π^*) such that
   - i) $y^* \neq y$ but
   - ii) $\mathsf{Ver}^R_{\mathsf{pp}}(x, \pi^*, y^*) = 1$



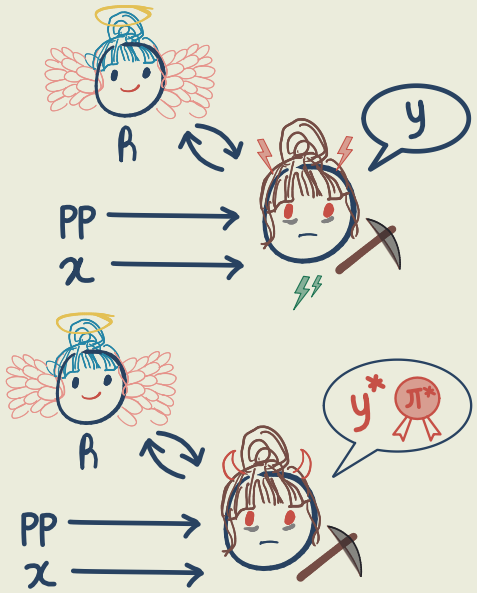
RO-Based VDF

is a triple of oracle algorithms $(\mathsf{Setup}^{\cdot}, \mathsf{Eval}^{\cdot}, \mathsf{Ver}^{\cdot})$ with following syntax:



Security requirements:

1. Sequentiality: no $\ll T$-round oracle adversary, given $x \leftarrow \mathcal{X}$, can produce $y := f(x)$, even with preprocessing
2. ~~Perfect~~ Comp. uniqueness: no PPT adversary, given $x \leftarrow \mathcal{X}$, can produce (y^*, π^*) such that
   - i) $y^* \neq y$ but
   - ii) $\mathsf{Ver}^R_{\mathsf{pp}}(x, \pi^*, y^*) = 1$



Breaking RO-Based VDF



Work	Restrictions on VDF	⚠ What is Broken?
[MSW20]	Perfect uniqueness & Public setup	Sequentiality

Breaking RO-Based VDF



Work	Restrictions on VDF	What is Broken?
[MSW20]	Perfect uniqueness & Public setup	Sequentiality
[DGMV20]	Tight sequentiality	Uniqueness

Breaking RO-Based VDF



Work	Restrictions on VDF	What is Broken?
[MSW20]	Perfect uniqueness & Public setup	Sequentiality
[DGMV20]	Tight sequentiality	Uniqueness
[GRY25]	Public setup	Sequentiality or uniqueness

Breaking RO-Based VDF



Work	Restrictions on VDF	What is Broken?
[MSW20]	Perfect uniqueness & Public setup	Sequentiality
[DGMV20]	Tight sequentiality	Uniqueness
[GRY25]	Public setup	Sequentiality or uniqueness
[AAK+26]	-	Sequentiality or uniqueness

Breaking RO-Based VDF



Work	Restrictions on VDF	⚠ What is Broken?
[MSW20]	Perfect uniqueness & Public setup	Sequentiality
[DGMV20]	Tight sequentiality	Uniqueness
[GRY25]	Public setup	Sequentiality or uniqueness
[AAK+26]	-	Sequentiality or uniqueness

How to Break Sequentiality?...

Assumptions: perfect sequentiality, public set-up and deterministic Eval

How to Break Sequentiality?...

- Based on ideas from Rudich's separation:
  - OWF/RO $\not\Rightarrow$ OWP
  - More generally: "imperfect" primitive $\not\Rightarrow$ "perfect" primitive

Assumptions: perfect sequentiality, public set-up and deterministic Eval

How to Break Sequentiality?...

- Based on ideas from Rudich's separation:
  - OWF/RO $\not\Rightarrow$ OWP
  - More generally: "imperfect" primitive $\not\Rightarrow$ "perfect" primitive



Main idea: given pp and x, it is possible to learn all R-queries made in $\mathsf{Ver}^{R}_{\mathtt{pp}}(x, \pi, y)$ in $\approx t$ rounds

---

Assumptions: perfect sequentiality, public set-up and deterministic Eval

How to Break Sequentiality?...

- Based on ideas from Rudich's separation:
  - OWF/RO $\not\Rightarrow$ OWP
  - More generally: "imperfect" primitive $\not\Rightarrow$ "perfect" primitive



Main idea: given pp and x, it is possible to learn all R-queries made in $\mathsf{Ver}^{R}_{\mathsf{pp}}(x, \pi, y)$ in $\approx t$ rounds



Key tool: hybrid of oracles

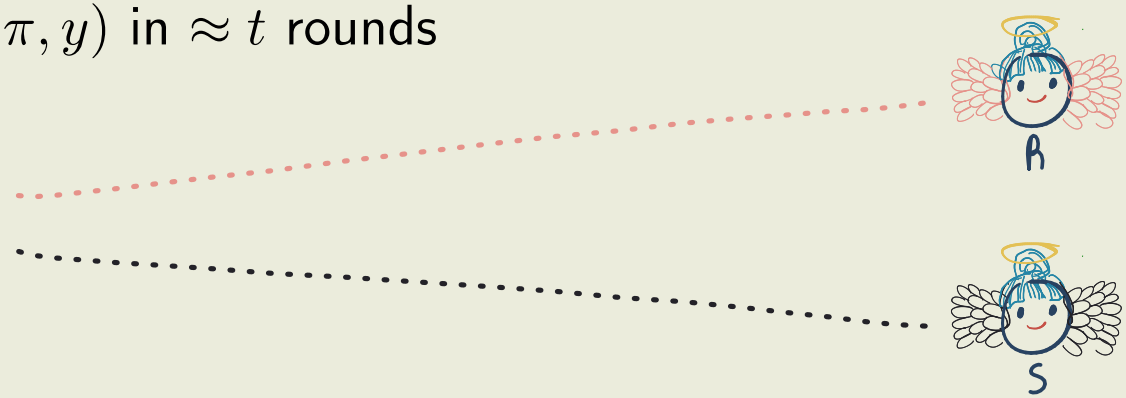
Assumptions: perfect sequentiality, public set-up and deterministic Eval

How to Break Sequentiality?...

- Based on ideas from Rudich's separation:
  - OWF/RO $\not\Rightarrow$ OWP
  - More generally: "imperfect" primitive $\not\Rightarrow$ "perfect" primitive



Main idea: given pp and x, it is possible to learn all R-queries made in $\mathsf{Ver}^{R}_{\mathsf{pp}}(x, \pi, y)$ in $\approx t$ rounds





Key tool: hybrid of oracles R and S at $\mathcal{Q} \subseteq \{0,1\}^*$ is defined as

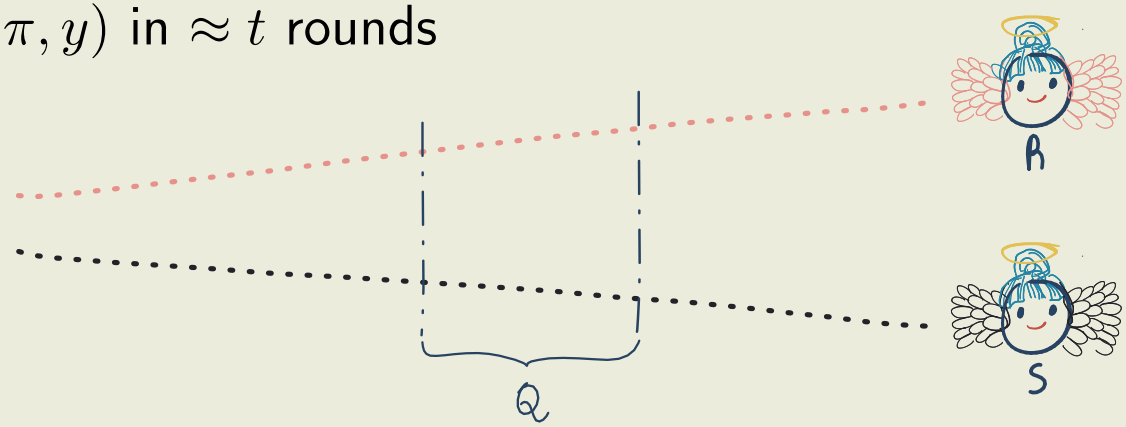
$$H(q) := \begin{cases} S(q) & \text{if } q \in \mathcal{Q} \\ R(q) & \text{otherwise} \end{cases}$$

Assumptions: perfect sequentiality, public set-up and deterministic Eval

How to Break Sequentiality?...

- Based on ideas from Rudich's separation:
  - OWF/RO $\not\Rightarrow$ OWP
  - More generally: "imperfect" primitive $\not\Rightarrow$ "perfect" primitive

Main idea: given pp and x, it is possible to learn all R-queries made in $\mathsf{Ver}^{R}_{\mathsf{pp}}(x, \pi, y)$ in $\approx t$ rounds



Key tool: hybrid of oracles R and S at $\mathcal{Q} \subseteq \{0,1\}^*$ is defined as

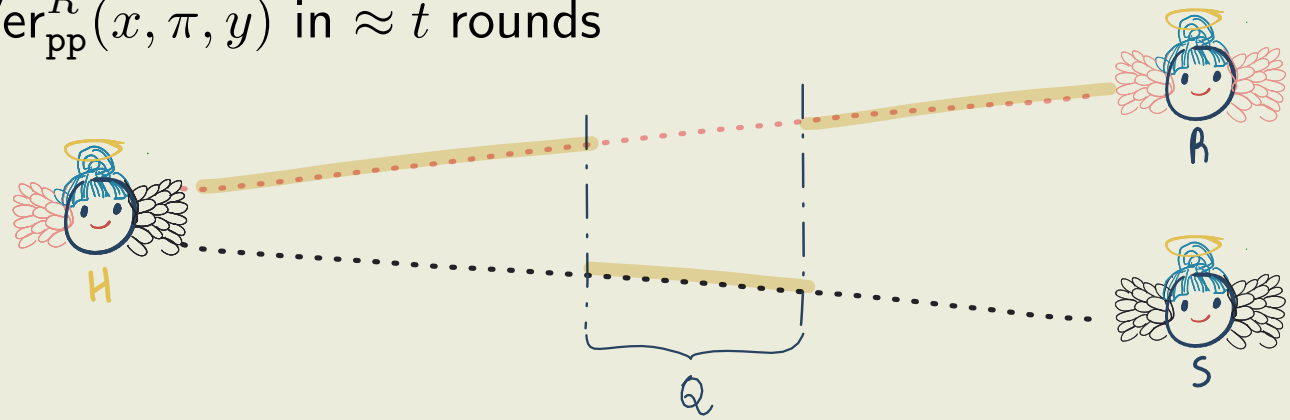
$$H(q) := \begin{cases} S(q) & \text{if } q \in \mathcal{Q} \\ R(q) & \text{otherwise} \end{cases}$$

Assumptions: perfect sequentiality, public set-up and deterministic Eval

How to Break Sequentiality?...

- Based on ideas from Rudich's separation:
  - OWF/RO $\not\Rightarrow$ OWP
  - More generally: "imperfect" primitive $\not\Rightarrow$ "perfect" primitive

Main idea: given pp and x, it is possible to learn all R-queries made in $\mathsf{Ver}^{R}_{\mathsf{pp}}(x, \pi, y)$ in $\approx t$ rounds

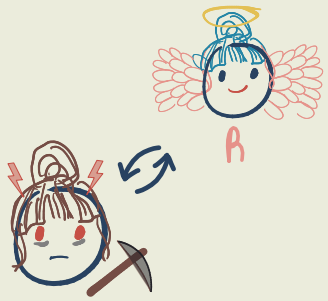


Key tool: hybrid of oracles R and S at $\mathcal{Q} \subseteq \{0,1\}^*$ is defined as

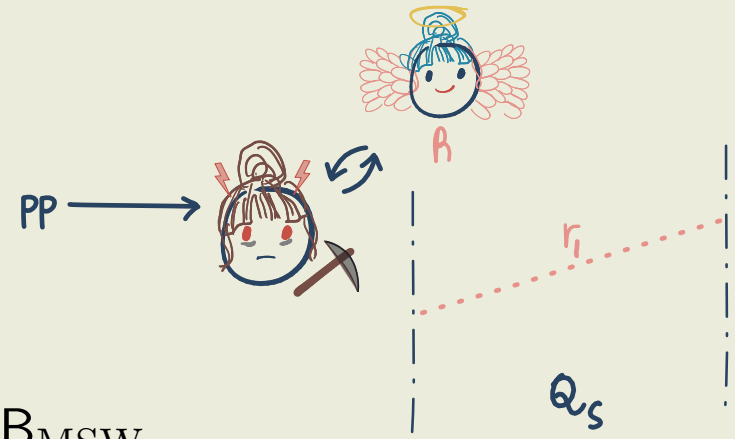
$$H(q) := \begin{cases} S(q) & \text{if } q \in \mathcal{Q} \\ R(q) & \text{otherwise} \end{cases}$$

Assumptions: perfect sequentiality, public set-up and deterministic Eval

How to Break Sequentiality?...



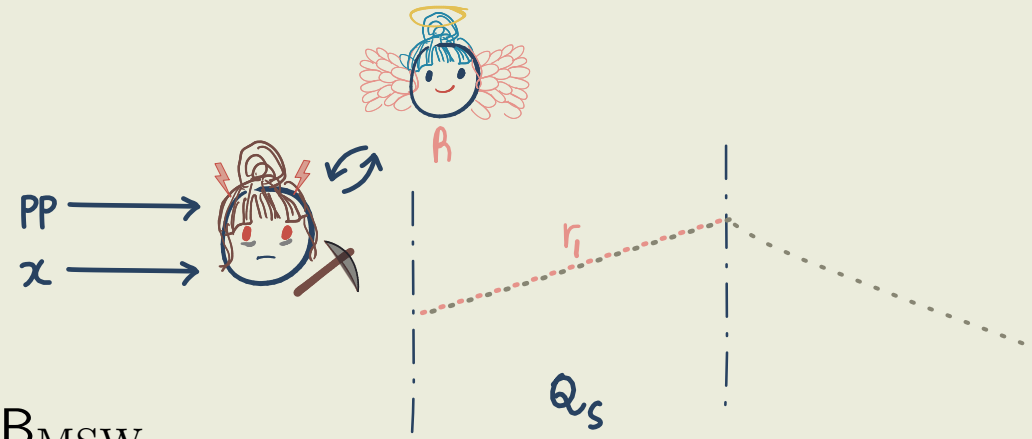
How to Break Sequentiality?...



$\mathsf{B}_{\mathrm{MSW}}$

1: Initialise *partial* function r_1 to equal R at $\mathcal{Q}_S$ ← R-queries by Setup

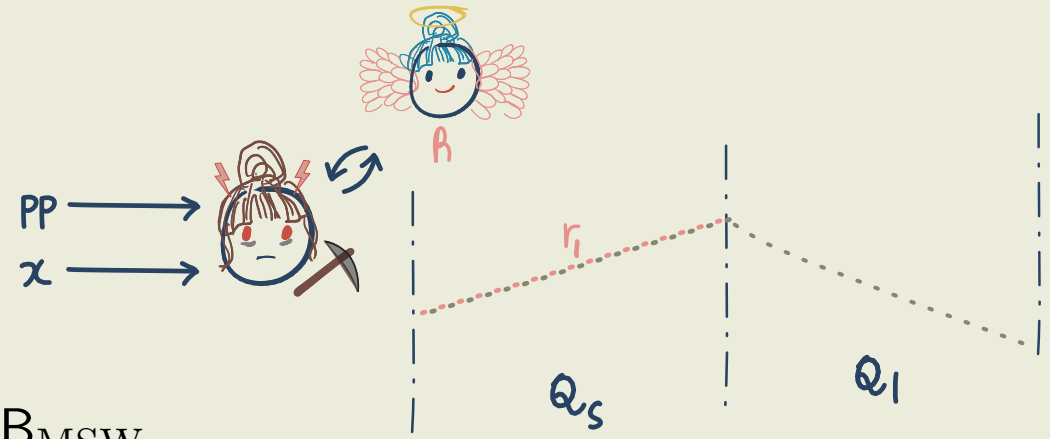
How to Break Sequentiality?...



$\mathsf{B}_{\mathrm{MSW}}$

1: Initialise *partial* function r_1 to equal R at $\mathcal{Q}_S$ ← R-queries by Setup
2: **for** $i \in [1, 2t]$ **do**
3: Sample a simulated oracle S_i that "extends" r_i
4: Compute $(y_i, \pi_i) := \mathsf{Eval}_{\mathsf{pp}}^{S_i}(x)$ and let $\mathcal{Q}_i$ be "new" S_i-queries

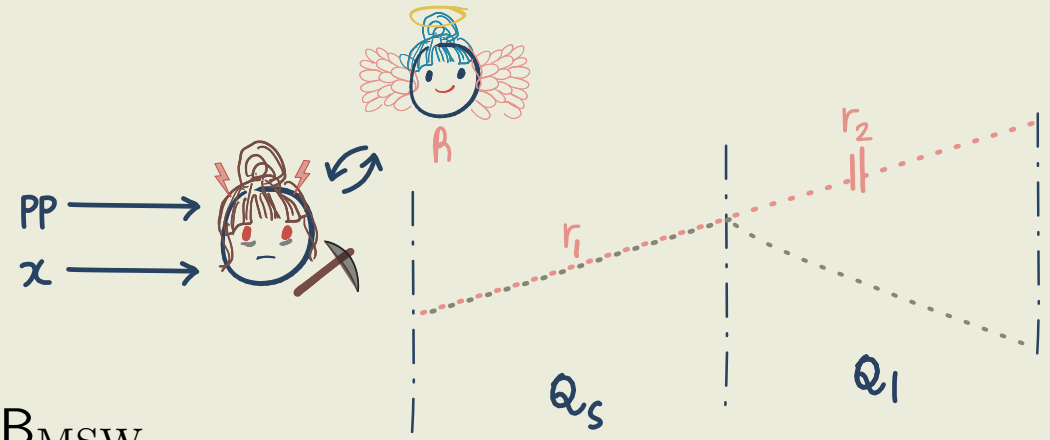
How to Break Sequentiality?...



$\mathsf{B}_{\mathrm{MSW}}$

1: Initialise *partial* function r_1 to equal R at $\mathcal{Q}_S$ ← R-queries by Setup
2: **for** $i \in [1, 2t]$ **do**
3: Sample a simulated oracle S_i that "extends" r_i
4: Compute $(y_i, \pi_i) := \mathsf{Eval}_{\mathsf{pp}}^{S_i}(x)$ and let $\mathcal{Q}_i$ be "new" S_i-queries

How to Break Sequentiality?...



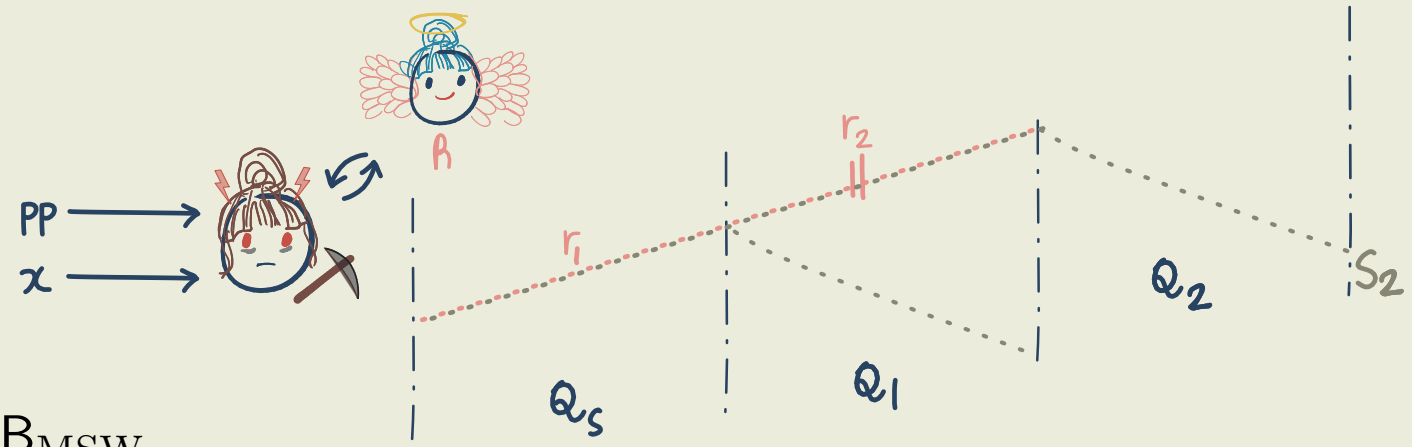
$\mathsf{B}_{\mathrm{MSW}}$

1: Initialise *partial* function r_1 to equal R at $\mathcal{Q}_S$ ← R-queries by Setup
2: **for** $i \in [1, 2t]$ **do**
3: Sample a simulated oracle S_i that "extends" r_i
4: Compute $(y_i, \pi_i) := \mathsf{Eval}_{\mathsf{pp}}^{S_i}(x)$ and let $\mathcal{Q}_i$ be "new" S_i-queries
5: Update partial function by querying R at $\mathcal{Q}_i$ *in parallel*:

$$r_{i+1}(q) := \begin{cases} r_i(q) & \text{if } q \text{ already defined in } r_i \\ R(q) & \text{if } q \in \mathcal{Q}_i \end{cases}$$

6: **end for**

How to Break Sequentiality?...



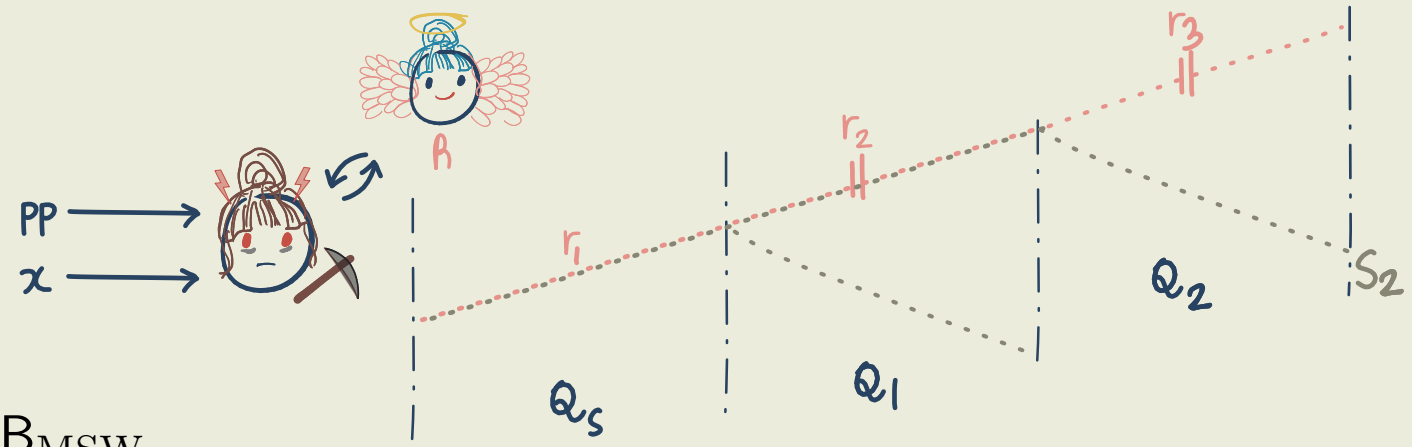
$\mathsf{B}_{\mathrm{MSW}}$

1: Initialise *partial* function r_1 to equal R at $\mathcal{Q}_S$ ← R-queries by Setup
2: **for** $i \in [1, 2t]$ **do**
3: Sample a simulated oracle S_i that "extends" r_i
4: Compute $(y_i, \pi_i) := \mathsf{Eval}_{\mathsf{pp}}^{S_i}(x)$ and let $\mathcal{Q}_i$ be "new" S_i-queries
5: Update partial function by querying R at $\mathcal{Q}_i$ *in parallel*:

$$r_{i+1}(q) := \begin{cases} r_i(q) & \text{if } q \text{ already defined in } r_i \\ R(q) & \text{if } q \in \mathcal{Q}_i \end{cases}$$

6: **end for**

How to Break Sequentiality?...



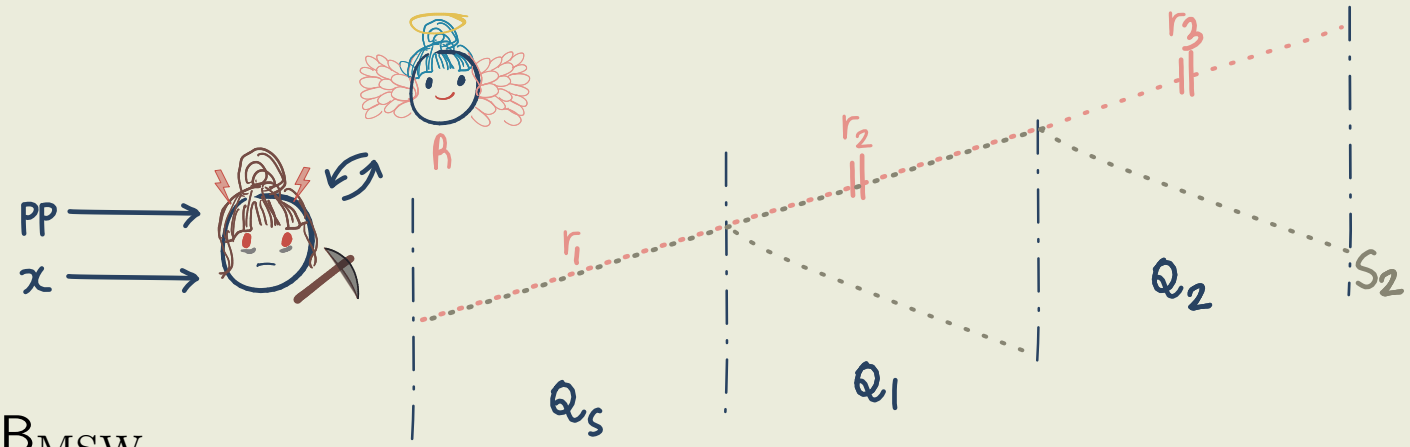
$\mathsf{B}_{\mathrm{MSW}}$

1: Initialise *partial* function r_1 to equal R at $\mathcal{Q}_S$ ← R-queries by Setup
2: **for** $i \in [1, 2t]$ **do**
3: Sample a simulated oracle S_i that "extends" r_i
4: Compute $(y_i, \pi_i) := \mathsf{Eval}_{\mathsf{pp}}^{S_i}(x)$ and let $\mathcal{Q}_i$ be "new" S_i-queries
5: Update partial function by querying R at $\mathcal{Q}_i$ *in parallel*:

$$r_{i+1}(q) := \begin{cases} r_i(q) & \text{if } q \text{ already defined in } r_i \\ R(q) & \text{if } q \in \mathcal{Q}_i \end{cases}$$

6: **end for**

How to Break Sequentiality?...



$\mathsf{B}_{\mathrm{MSW}}$

1: Initialise *partial* function r_1 to equal R at $\mathcal{Q}_S$ ← R-queries by Setup
2: **for** $i \in [1, 2t]$ **do**
3: Sample a simulated oracle S_i that "extends" r_i
4: Compute $(y_i, \pi_i) := \mathsf{Eval}_{\mathsf{pp}}^{S_i}(x)$ and let $\mathcal{Q}_i$ be "new" S_i-queries
5: Update partial function by querying R at $\mathcal{Q}_i$ *in parallel*:

$$r_{i+1}(q) := \begin{cases} r_i(q) & \text{if } q \text{ already defined in } r_i \\ R(q) & \text{if } q \in \mathcal{Q}_i \end{cases}$$

6: **end for**
7: Output y_{i^*} for $i^* \leftarrow [1, 2t]$

How to Break Sequentiality?...

Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

How to Break Sequentiality?...

Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

Proof.

- $\mathrm{B}_{\mathrm{MSW}}$ "learns" in round i if $\mathcal{Q}_i$ *contains at least one query* from $\mathcal{Q}_V$

$\mathcal{R}$-queries by $\mathsf{Verify}^{\mathcal{R}}_{\mathsf{pp}}(x, \pi, y)$

How to Break Sequentiality?...

Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

Proof.

- $\mathrm{B}_{\mathrm{MSW}}$ "learns" in round i if $\mathcal{Q}_i$ *contains at least one query* from $\mathcal{Q}_V$

1. Subclaim 1: There are *at most* t rounds in which $\mathrm{B}_{\mathrm{MSW}}$ learns

How to Break Sequentiality?...

Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

Proof.

- $\mathrm{B}_{\mathrm{MSW}}$ "learns" in round i if $\mathcal{Q}_i$ *contains at least one query* from $\mathcal{Q}_V$

1. Subclaim 1: There are *at most* t rounds in which $\mathrm{B}_{\mathrm{MSW}}$ learns
   - Since $|\mathcal{Q}_V| \leq t$ *and* R is queried on $\mathcal{Q}_i$ subsequently

How to Break Sequentiality?...

Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

Proof.

- $\mathrm{B}_{\mathrm{MSW}}$ "learns" in round i if $\mathcal{Q}_i$ *contains at least one query* from $\mathcal{Q}_V$

1. Subclaim 1: There are *at most* t rounds in which $\mathrm{B}_{\mathrm{MSW}}$ learns
   - Since $|\mathcal{Q}_V| \leq t$ *and* R is queried on $\mathcal{Q}_i$ subsequently
2. Subclaim 2: If $\mathrm{B}_{\mathrm{MSW}}$ does not learn in round i^* then $y_{i^*} = y$

How to Break Sequentiality?...

Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

Proof.

$\mathrm{B}_{\mathrm{MSW}}$ "learns" in round i if $\mathcal{Q}_i$ *contains at least one query* from $\mathcal{Q}_V$

1. Subclaim 1: There are *at most* t rounds in which $\mathrm{B}_{\mathrm{MSW}}$ learns
   - Since $|\mathcal{Q}_V| \leq t$ *and* R is queried on $\mathcal{Q}_i$ subsequently
2. Subclaim 2: If $\mathrm{B}_{\mathrm{MSW}}$ does not learn in round i^* then $y_{i^*} = y$
   - If $y_{i^*} \neq y$, then *uniqueness violated*

How to Break Sequentiality?...

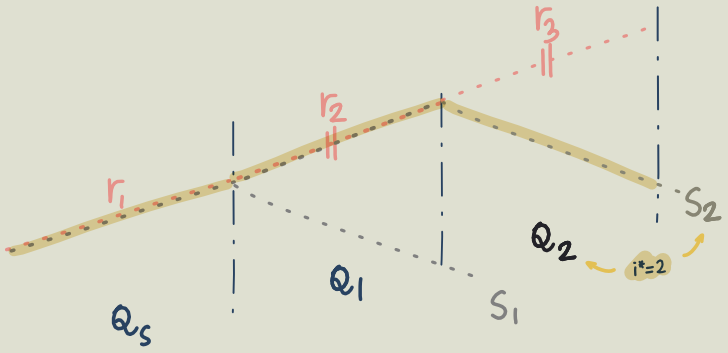
Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

Proof.

- $\mathrm{B}_{\mathrm{MSW}}$ "learns" in round i if $\mathcal{Q}_i$ *contains at least one query* from $\mathcal{Q}_V$

1. Subclaim 1: There are *at most* t rounds in which $\mathrm{B}_{\mathrm{MSW}}$ learns
   - Since $|\mathcal{Q}_V| \leq t$ *and* R is queried on $\mathcal{Q}_i$ subsequently
2. Subclaim 2: If $\mathrm{B}_{\mathrm{MSW}}$ does not learn in round i^* then $y_{i^*} = y$
   - If $y_{i^*} \neq y$, then *uniqueness violated* relative to the hybrid oracle



$$H(q) := \begin{cases} S_{i^*}(q) & \text{if } q \in \mathcal{Q}_{i^*} \\ R(q) & \text{otherwise} \end{cases}$$

How to Break Sequentiality?...

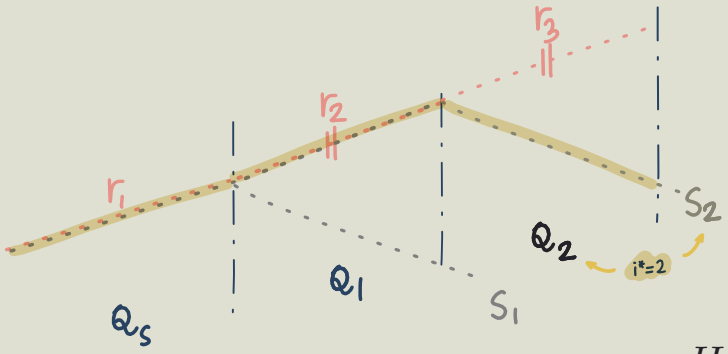
Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

Proof.

$\mathsf{B}_{\mathrm{MSW}}$ "learns" in round i if $\mathcal{Q}_i$ *contains at least one query* from $\mathcal{Q}_V$

1. Subclaim 1: There are *at most* t rounds in which $\mathsf{B}_{\mathrm{MSW}}$ learns
   - Since $|\mathcal{Q}_V| \leq t$ *and* R is queried on $\mathcal{Q}_i$ subsequently
2. Subclaim 2: If $\mathsf{B}_{\mathrm{MSW}}$ does not learn in round i^* then $y_{i^*} = y$
   - If $y_{i^*} \neq y$, then *uniqueness violated* relative to the hybrid oracle



$$H(q) := \begin{cases} S_{i^*}(q) & \text{if } q \in \mathcal{Q}_{i^*} \\ R(q) & \text{otherwise} \end{cases}$$

as $(y_{i^*}, \pi_{i^*}) = \mathsf{Eval}^H_{\mathsf{pp}}(x)$ and $\mathsf{Ver}^H_{\mathsf{pp}}(x, \pi, y) = 1$ □

How to Break Sequentiality?...

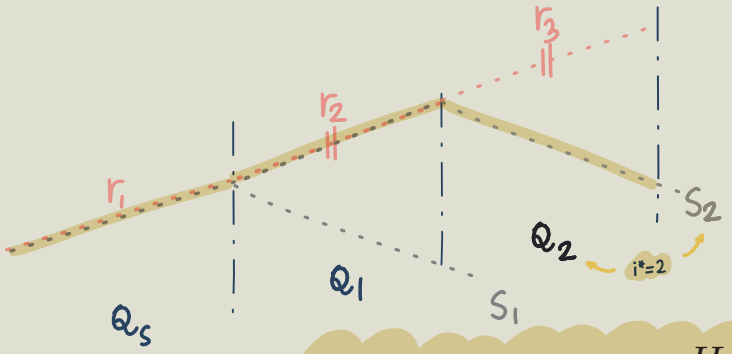
Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

Proof.

- $\mathrm{B}_{\mathrm{MSW}}$ "learns" in round i if $\mathcal{Q}_i$ *contains at least one query* from $\mathcal{Q}_V$

1. Subclaim 1: There are *at most* t rounds in which $\mathrm{B}_{\mathrm{MSW}}$ learns
   - Since $|\mathcal{Q}_V| \leq t$ *and* R is queried on $\mathcal{Q}_i$ subsequently
2. Subclaim 2: If $\mathrm{B}_{\mathrm{MSW}}$ does not learn in round i^* then $y_{i^*} = y$
   - If $y_{i^*} \neq y$, then *uniqueness violated* relative to the hybrid oracle



$$H(q) := \begin{cases} S_{i^*}(q) & \text{if } q \in \mathcal{Q}_{i^*} \\ R(q) & \text{otherwise} \end{cases}$$

as $(y_{i^*}, \pi_{i^*}) = \mathsf{Eval}^{H}_{\mathrm{pp}}(x)$ and $\mathsf{Ver}^{H}_{\mathrm{pp}}(x, \pi, y) = 1$ □

H consistent with S_{i^*}

How to Break Sequentiality?...

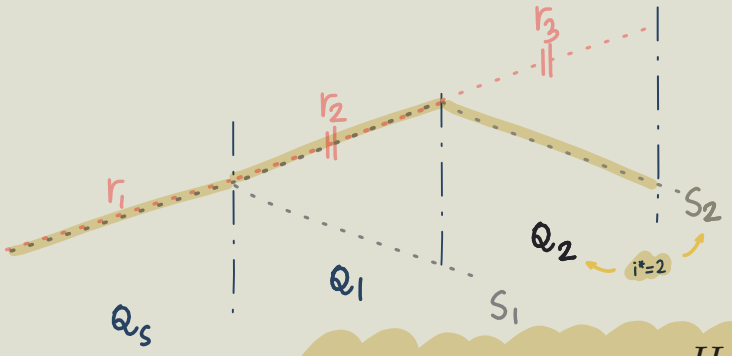
Claim 1

$y_{i^*} = y$ holds with probability at least $1/2$

Proof.

- $\mathrm{B}_{\mathrm{MSW}}$ "learns" in round i if $\mathcal{Q}_i$ *contains at least one query* from $\mathcal{Q}_V$

1. Subclaim 1: There are *at most* t rounds in which $\mathrm{B}_{\mathrm{MSW}}$ learns
   - Since $|\mathcal{Q}_V| \leq t$ *and* R is queried on $\mathcal{Q}_i$ subsequently
2. Subclaim 2: If $\mathrm{B}_{\mathrm{MSW}}$ does not learn in round i^* then $y_{i^*} = y$
   - If $y_{i^*} \neq y$, then *uniqueness violated* relative to the hybrid oracle

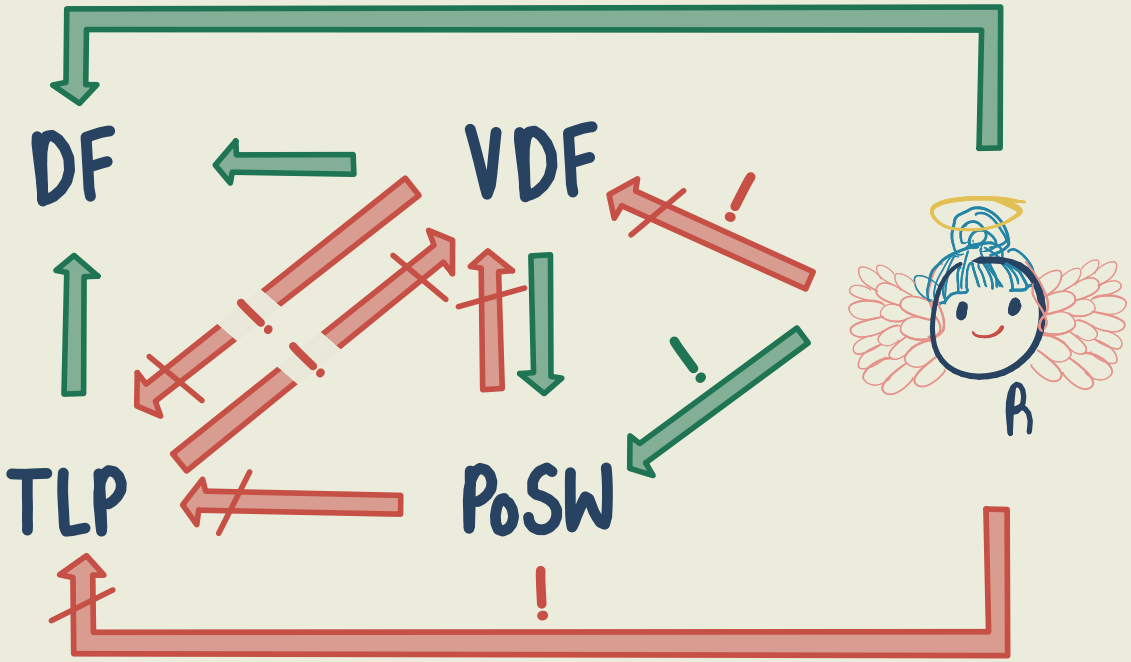


$$H(q) := \begin{cases} S_{i^*}(q) & \text{if } q \in \mathcal{Q}_{i^*} \\ R(q) & \text{otherwise} \end{cases}$$

as $(y_{i^*}, \pi_{i^*}) = \mathsf{Eval}^H_{\mathsf{pp}}(x)$ and $\mathsf{Ver}^H_{\mathsf{pp}}(x, \pi, y) = 1$ □

H consistent with S_{i^*} ↗ ↖ $\mathcal{Q}_{i^*} \cap \mathcal{Q}_V = \emptyset$

In Summary



ePrint 2025/1773


Hamza Abusalah, Nivesh Aggarwal, Karen Azari, Chethan Kamath, and Maximilian von Consbruch.

Separating verifiable delay functions and time-lock puzzles.

In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part V*, volume 16545 of *LNCS*, pages 33–63. Springer, Cham, May 2026.


Hamza Abusalah, Karen Azari, Chethan Kamath, Erkan Tairi, and Maximilian von Consbruch.

Impossibility of VDFs in the ROM: The complete picture.

In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part V*, volume 16545 of *LNCS*, pages 3–32. Springer, Cham, May 2026.


Arasu Arun, Joseph Bonneau, and Jeremy Clark.

Short-lived zero-knowledge proofs and signatures.

In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part III*, volume 13793 of *LNCS*, pages 487–516. Springer, Cham, December 2022.


Shweta Agrawal, Andrea Basso, and Sikhar Patranabis.

Post-quantum time-lock puzzle from isogenies.

Cryptology ePrint Archive, Paper 2026/2306, 2026.


Shweta Agrawal, Giulio Malavolta, and Tianwei Zhang.

Time-lock puzzles from lattices.

In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 425–456. Springer, Cham, August 2024.



Jeffrey Burdges and Luca De Feo.

Delay encryption.

In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 302–326. Springer, Cham, October 2021.



Richard Cleve.

Limits on the security of coin flips when half the processors are faulty (extended abstract).

In *18th ACM STOC*, pages 364–369. ACM Press, May 1986.



J. . Cai, R. J. Lipton, R. Sedgewick, and A. C. . Yao.

Towards uncheatable benchmarks.

In *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 2–11, May 1993.



Bram Cohen and Krzysztof Pietrzak.

Simple proofs of sequential work.

In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 451–467. Springer, Cham, April / May 2018.



Nico Döttling, Sanjam Garg, Giulio Malavolta, and Prashant Nalini Vasudevan.

Tight verifiable delay functions.

In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20*, volume 12238 of *LNCS*, pages 65–84. Springer, Cham, September 2020.



Cody Freitag, Ilan Komargodski, Rafael Pass, and Naomi Sirkin.

Non-malleable time-lock puzzles and applications.

In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 447–479. Springer, Cham, November 2021.



Matthew K. Franklin and Dahlia Malkhi.

Auditable metering with lightweight security.

In Rafael Hirschfeld, editor, *FC'97*, volume 1318 of *LNCS*, pages 151–160. Springer, Berlin, Heidelberg, February 1997.



Ziyi Guan, Artur Riazanov, and Weiqiang Yuan.

Breaking verifiable delay functions in the random oracle model.

In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 161–191. Springer, Cham, August 2025.



David M. Goldschlag and Stuart G. Stubblebine.

Publicly verifiable lotteries: Applications of delaying functions.

In Rafael Hirschfeld, editor, *FC'98*, volume 1465 of *LNCS*, pages 214–226. Springer, Berlin, Heidelberg, February 1998.



Russell W. F. Lai and Giulio Malavolta.

Subvector commitments with application to succinct arguments.

In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560. Springer, Cham, August 2019.



Arjen K. Lenstra and Benjamin Wesolowski.

Trustworthy public randomness with sloth, unicorn, and trx.

Int. J. Appl. Cryptogr., 3(4):330–343, 2017.



Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan.

Time-lock puzzles in the random oracle model.

In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 39–50. Springer, Berlin, Heidelberg, August 2011.



Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan.

Publicly verifiable proofs of sequential work.

In Robert D. Kleinberg, editor, *ITCS 2013*, pages 373–388. ACM, January 2013.



Mohammad Mahmoody, Caleb Smith, and David J. Wu.

Can verifiable delay functions be based on random oracles?

In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *ICALP 2020*, volume 168 of *LIPIcs*, pages 83:1–83:17. Schloss Dagstuhl, July 2020.



Krzysztof Pietrzak.

Simple verifiable delay functions.

In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15. LIPIcs, January 2019.



R. L. Rivest, A. Shamir, and D. A. Wagner.

Time-lock puzzles and timed-release crypto.

Technical report, Cambridge, MA, USA, 1996.