A Map of Witness Maps: New Definitions and Connections

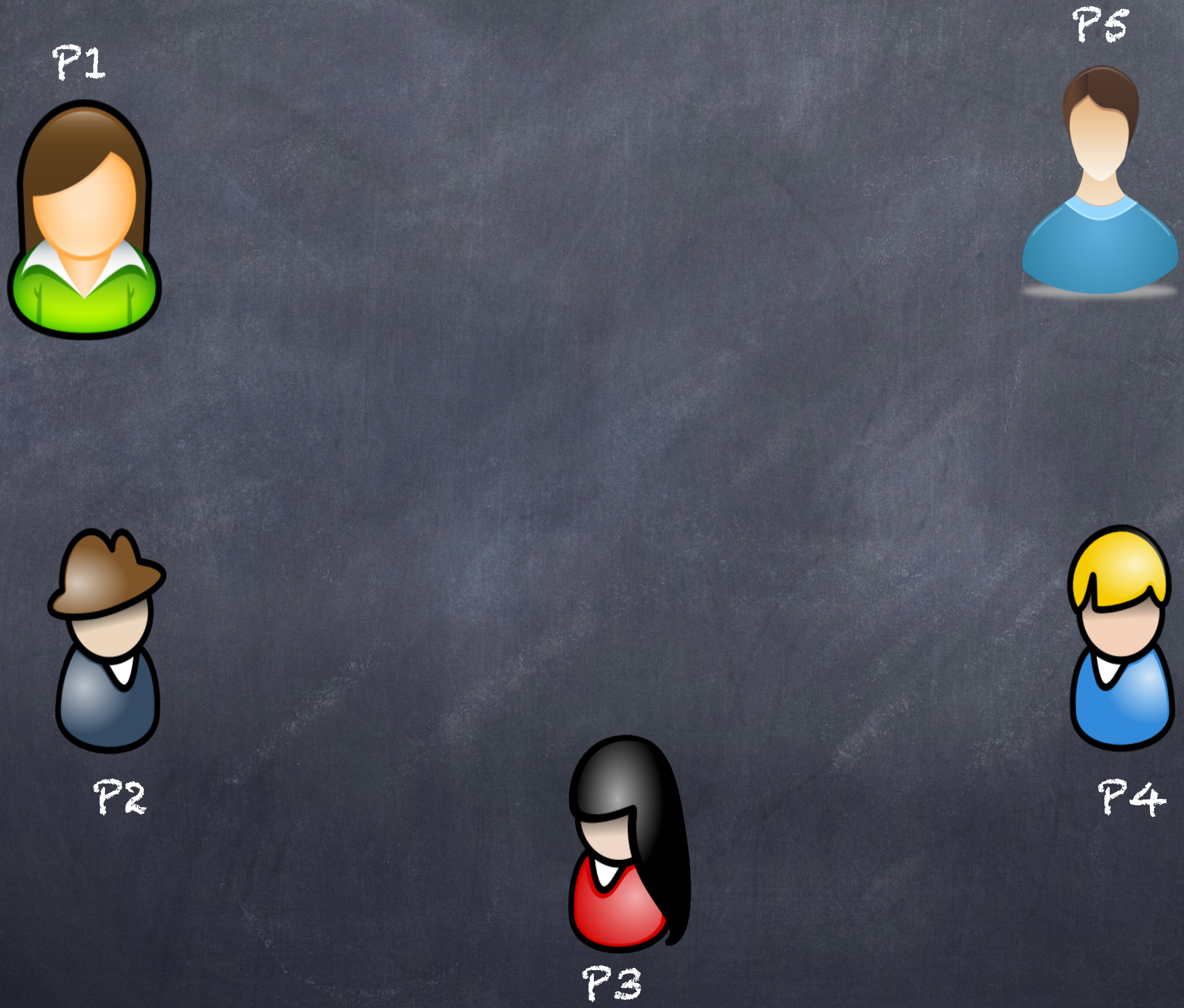
Suvradip Chakraborty*, Manoj Prabhakaran[2], Daniel Wichs[3]

[1]Visa Research [2]IIT Bombay
[3]Northeastern University & NTT Research

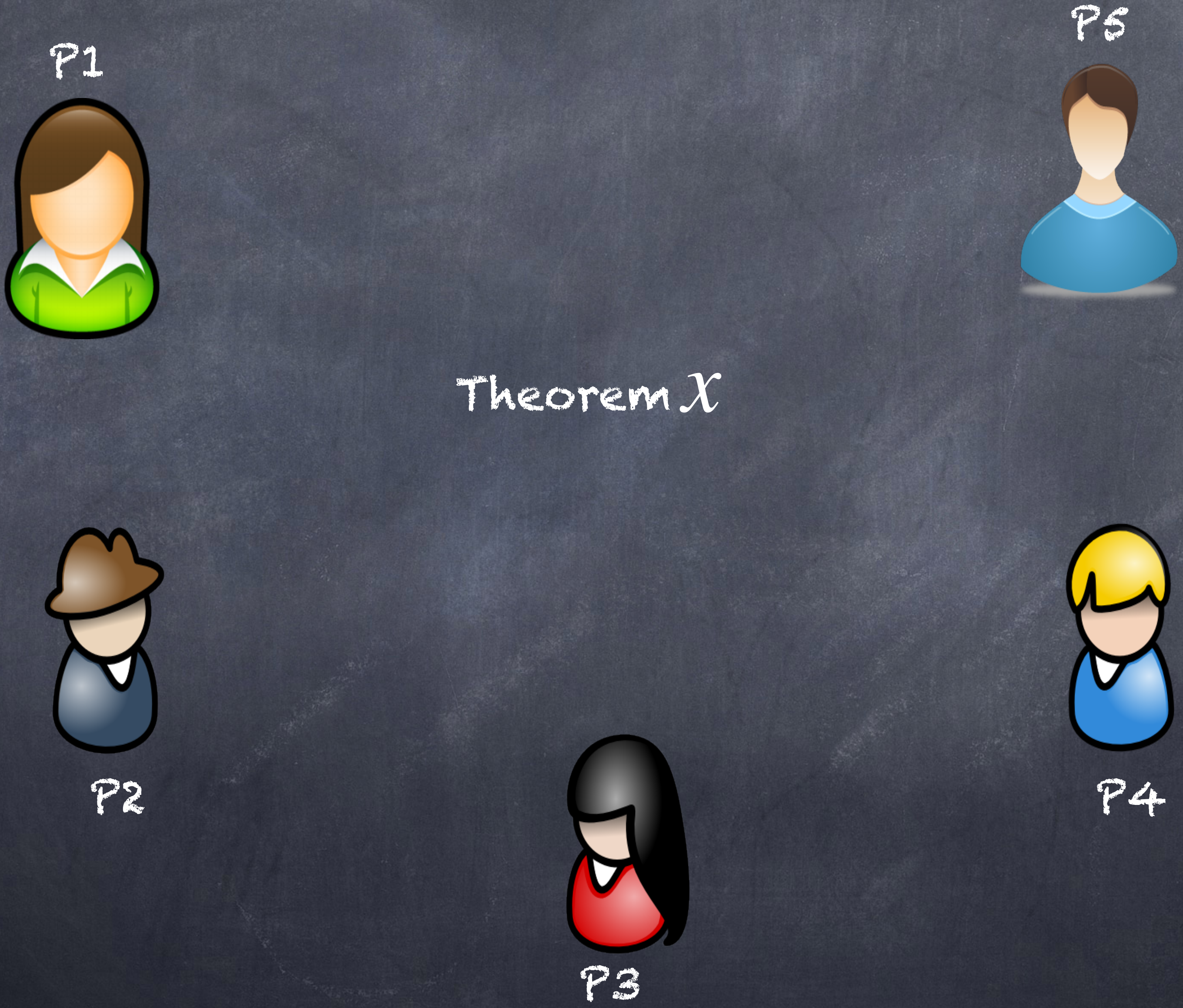
IACR PKC 2023

(Non)uniqueness of proofs

(Non)uniqueness of proofs

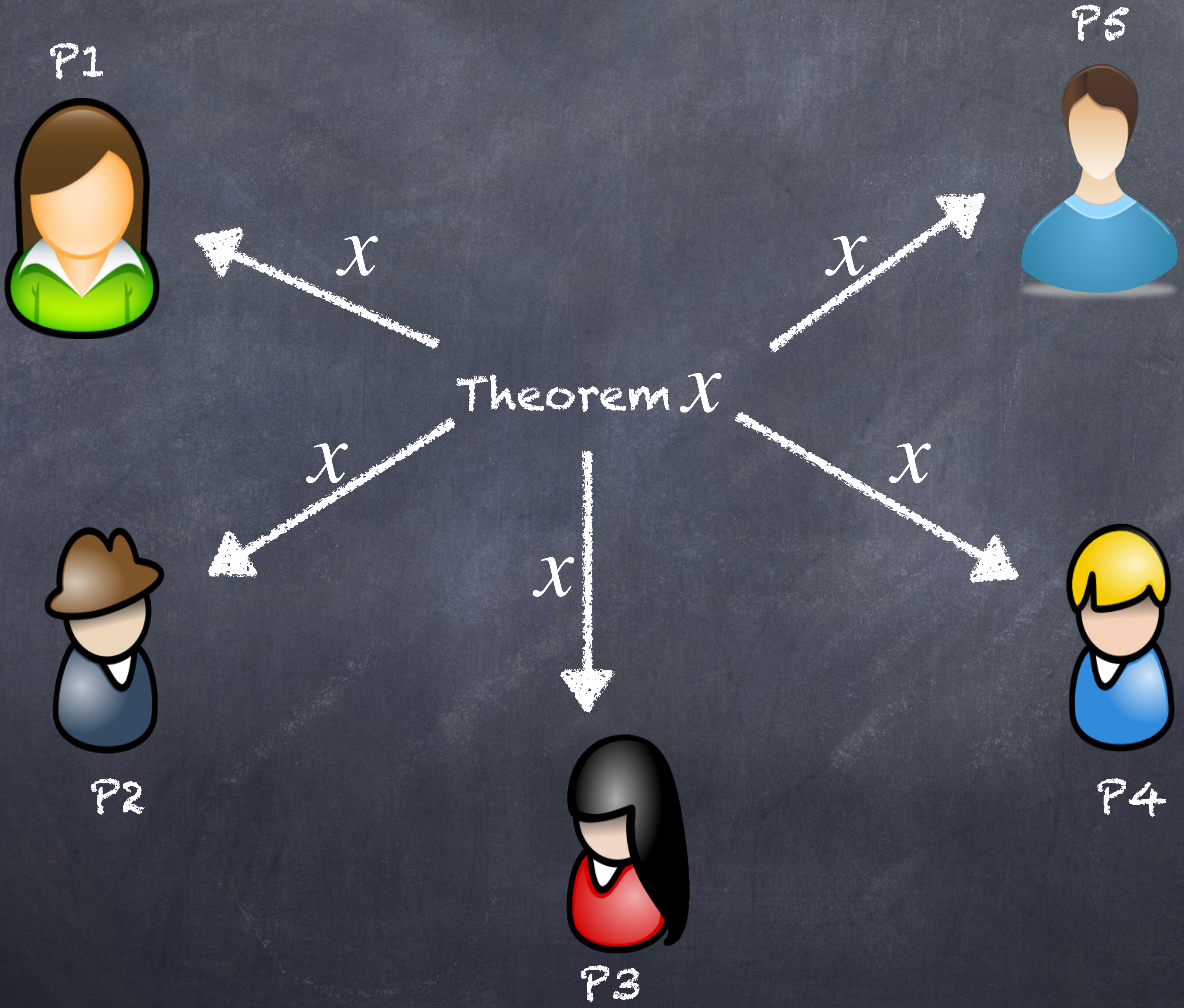


(Non)uniqueness of proofs



Theorem X

(Non)uniqueness of proofs



P1

P5

x

x

Theorem X

x

x

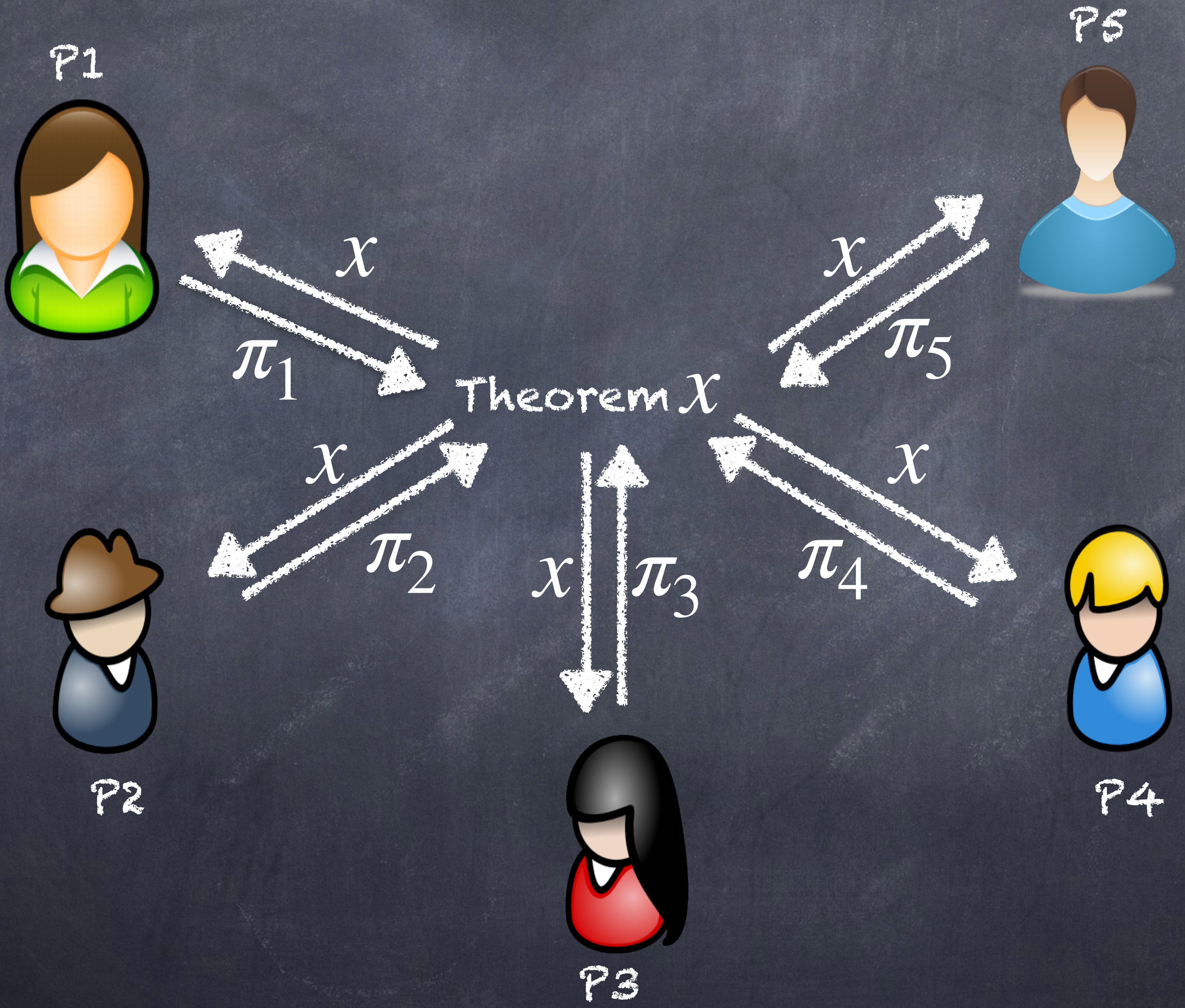
x

P2

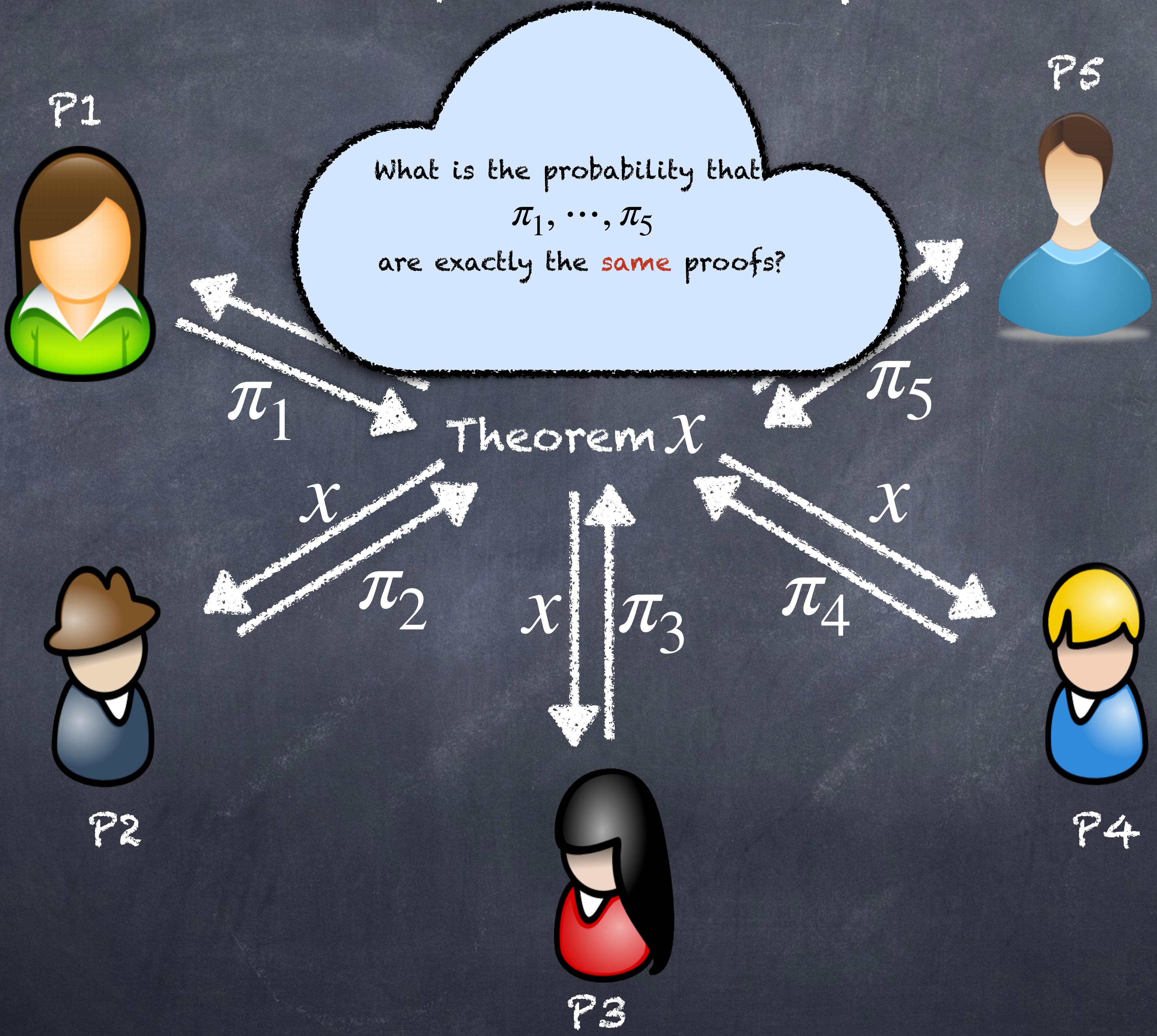
P3

P4

(Non)uniqueness of proofs

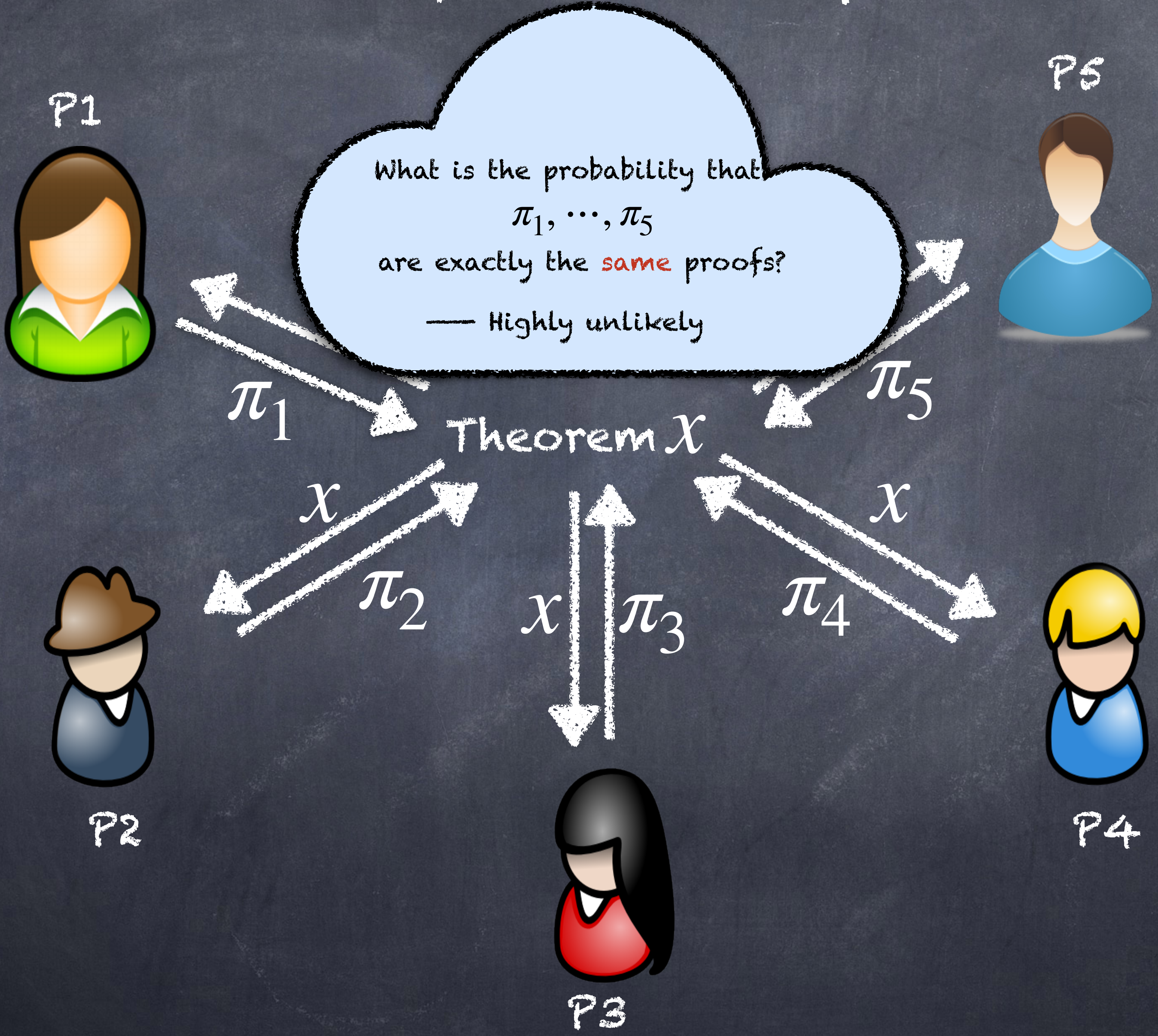


(Non)uniqueness of proofs



What is the probability that $\pi_1, \cdots, \pi_5$ are exactly the same proofs?

(Non)uniqueness of proofs



What is the probability that

$\pi_1, \cdots, \pi_5$

are exactly the same proofs?

—— Highly unlikely

Complexity class NP vs UP

Complexity class NP vs UP

- For NP languages a true statement (e.g., a graph is 3-colourable) will often multiple witnesses (many different 3-colourings).

Complexity class NP vs UP

- For NP languages a true statement (e.g., a graph is 3-colourable) will often multiple witnesses (many different 3-colourings).
- Can we come up with a proof system for NP languages where the proofs are guaranteed to be unique?

Complexity class NP vs UP

- For NP languages a true statement (e.g., a graph is 3-colourable) will often multiple witnesses (many different 3-colourings).
- Can we come up with a proof system for NP languages where the proofs are guaranteed to be unique?
  - Complexity class UP

Complexity class NP vs UP

- For NP languages a true statement (e.g., a graph is 3-colourable) will often multiple witnesses (many different 3-colourings).
- Can we come up with a proof system for NP languages where the proofs are guaranteed to be unique?
  - Complexity class UP
- Is NP= UP?

Complexity class NP vs UP

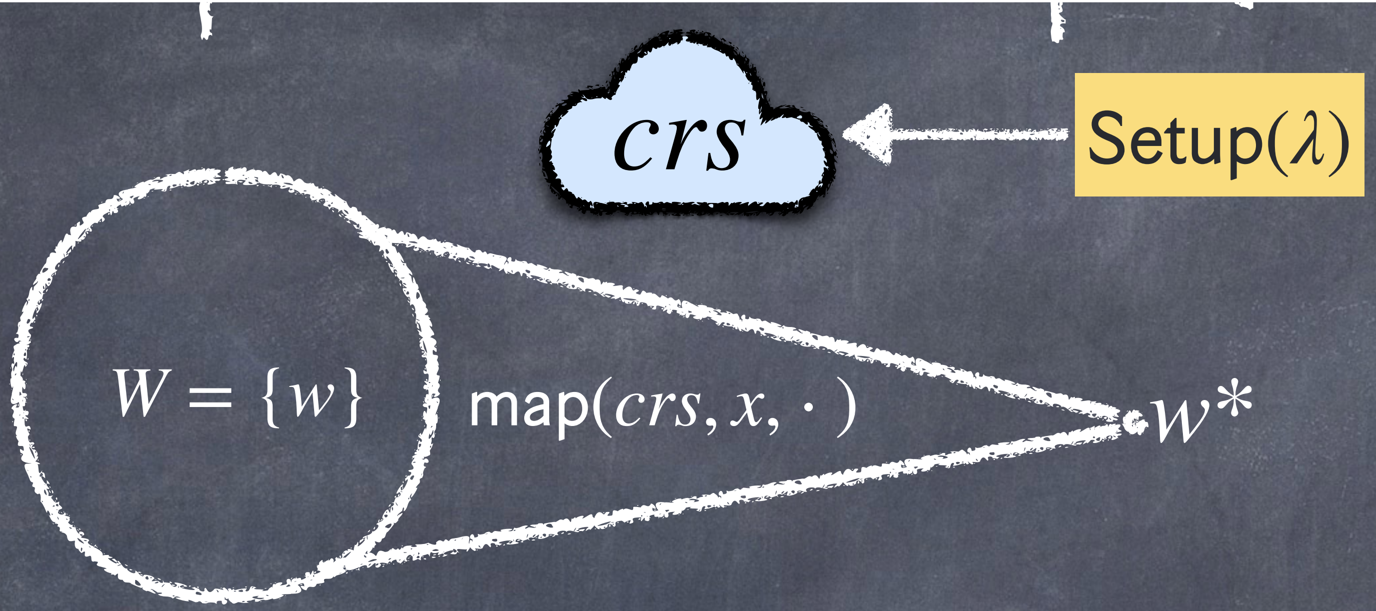
- For NP languages a true statement (e.g., a graph is 3-colourable) will often multiple witnesses (many different 3-colourings).
- Can we come up with a proof system for NP languages where the proofs are guaranteed to be unique?
    - Complexity class UP
- Is NP= UP?
  - Unlikely

Complexity class NP vs UP

- For NP languages a true statement (e.g., a graph is 3-colourable) will often multiple witnesses (many different 3-colourings).
- Can we come up with a proof system for NP languages where the proofs are guaranteed to be unique?
    - Complexity class UP
- Is NP= UP?
  - Unlikely
  - There exists oracle separations

Unique Witness Maps (UWM)

Unique Witness Maps (UWM)



crs

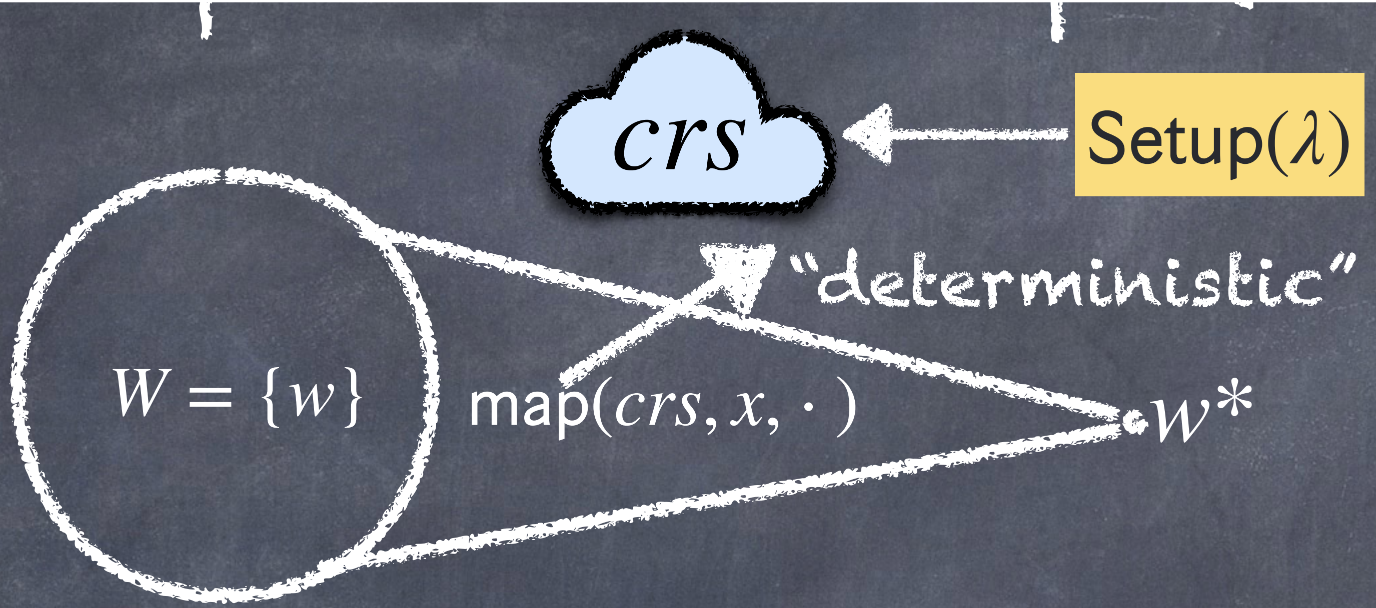
Setup(λ)

$W = \{w\}$

map($crs, x, \cdot$)

w^*

Unique Witness Maps (UWM)



crs

Setup(λ)

"deterministic"

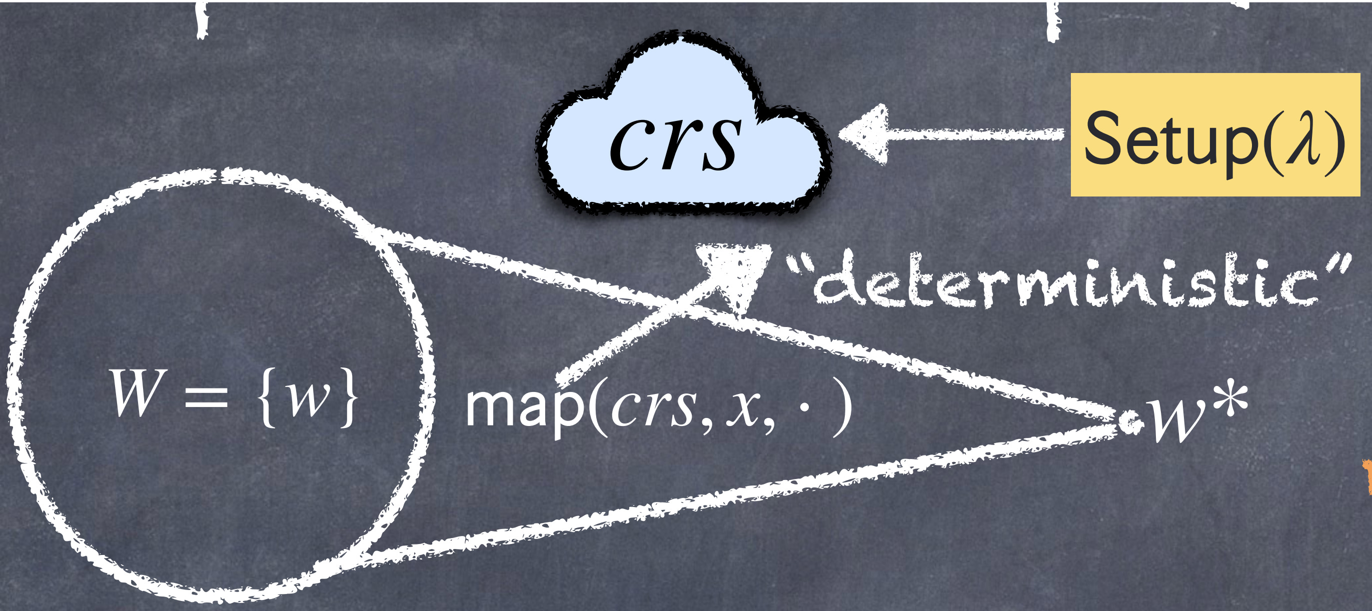
$W = \{w\}$

map$(crs, x, \cdot\ )$

w^*

Unique Witness Maps (UWM)

Witness set of $x \in \mathscr{L}$



crs

Setup(λ)

“deterministic”

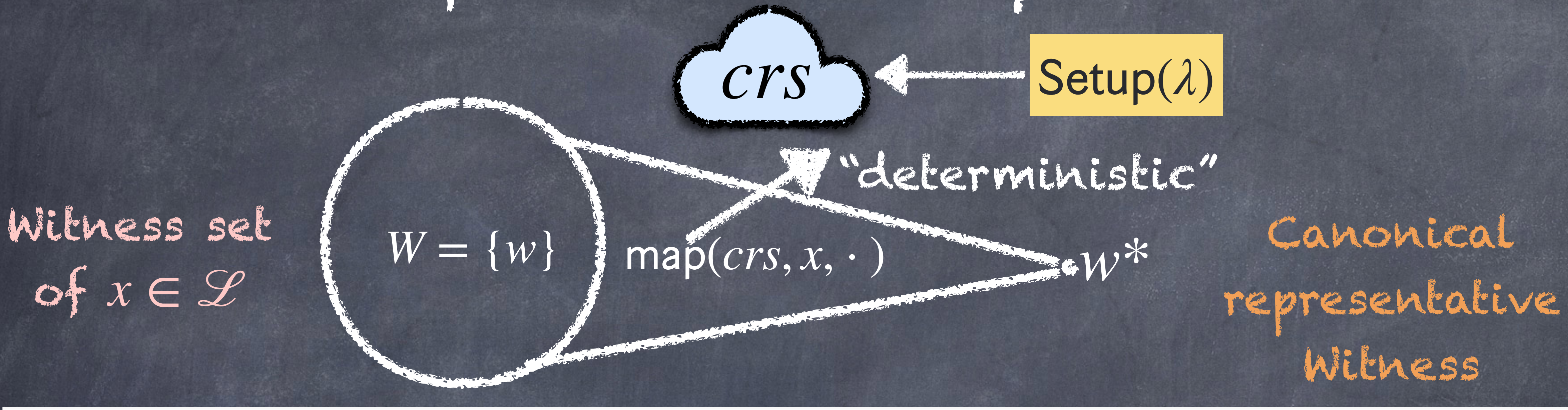
$W = \{w\}$

map$(crs, x, \cdot\ )$

w^*

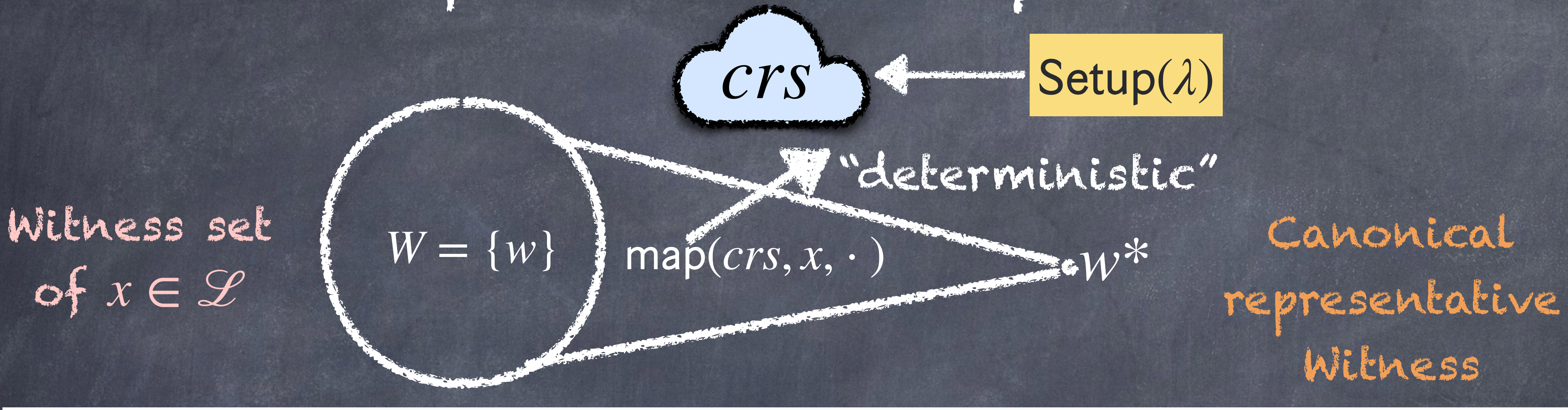
Canonical representative witness

Unique Witness Maps (UWM)



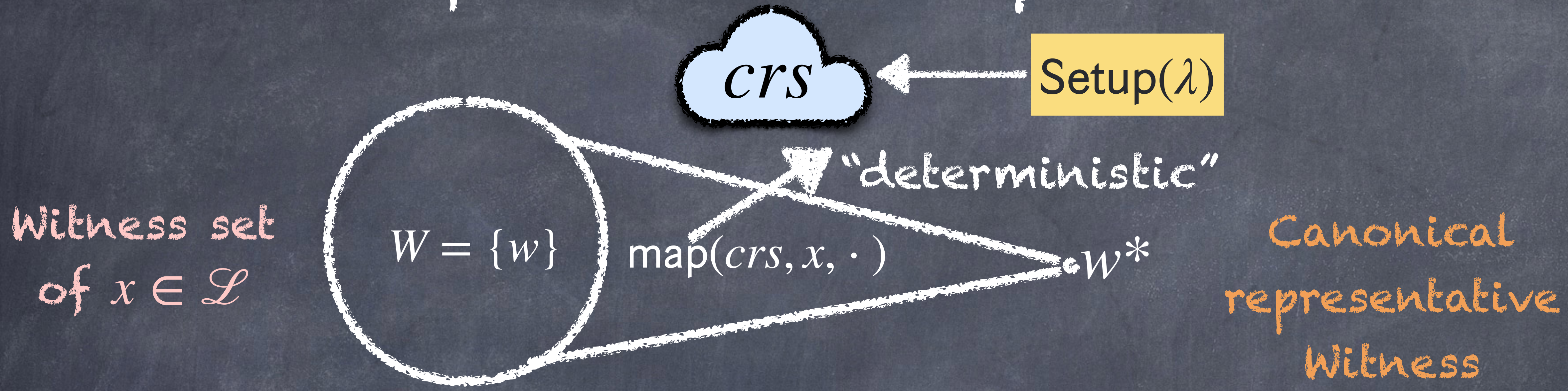
- Setup(λ): crs

Unique Witness Maps (UWM)



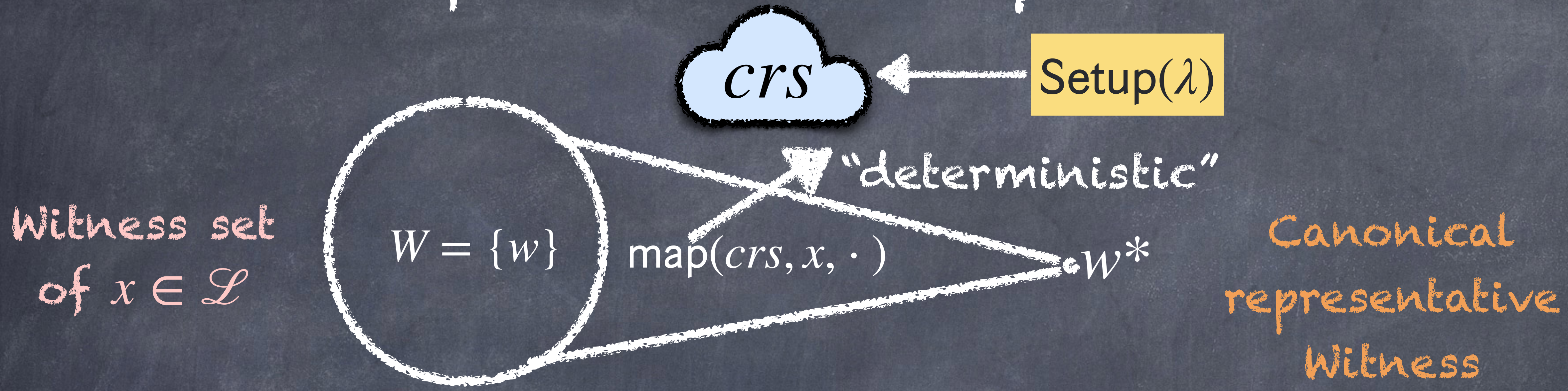
- Setup(λ): crs
- map(crs, x, w): w^*. The mapping is deterministic.

Unique Witness Maps (UWM)



- Setup(λ): crs
- map(crs, x, w): w^*. The mapping is deterministic.
- check(crs, x, w^*): 1/0

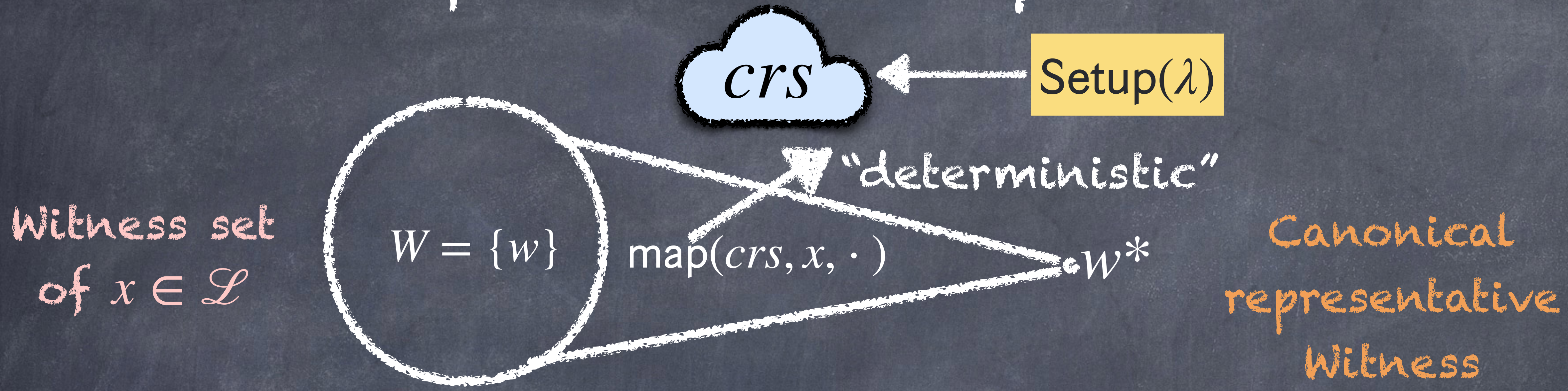
Unique Witness Maps (UWM)



- Setup(λ): crs
- map(crs, x, w): w^*. The mapping is deterministic.
- check(crs, x, w^*): 1/0

1. Completeness: If x is "true" ⤳ check o/p 1.

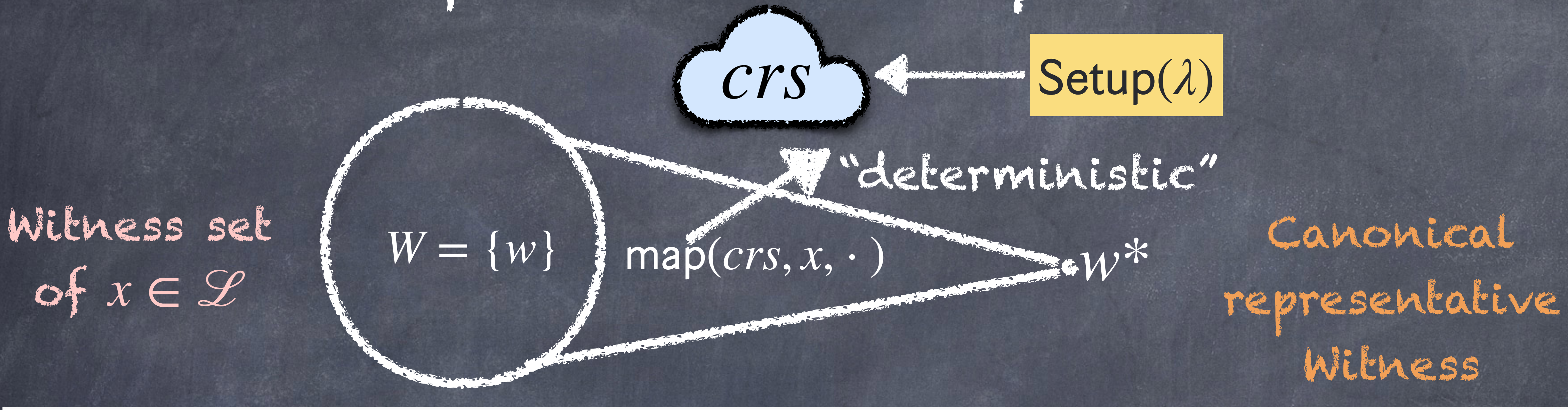
Unique Witness Maps (UWM)



- Setup(λ): crs
- map(crs, x, w): w^*. The mapping is deterministic.
- check(crs, x, w^*): 1/0

1. Completeness: If x is "true" ⇝ check o/p 1.
2. (Computational) Soundness: If x is "false" ⇝ check o/p 0.

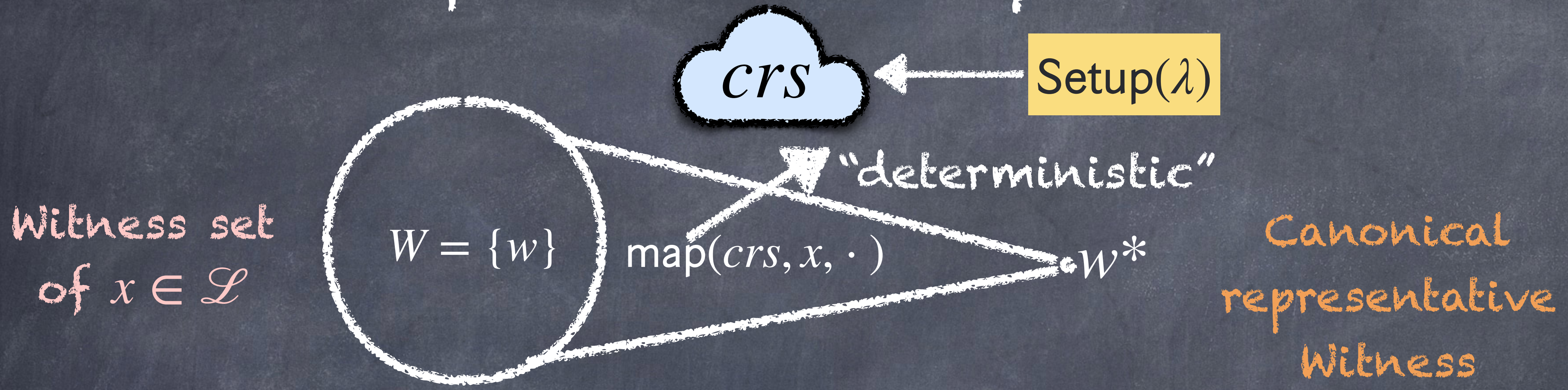
Unique Witness Maps (UWM)



- Setup(λ): crs
- map(crs, x, w): w^*. The mapping is deterministic.
- check(crs, x, w^*): 1/0

1. Completeness: If x is "true" ⇝ check o/p 1.
2. (Computational) Soundness: If x is "false" ⇝ check o/p 0.
3. Uniqueness: Unique proof w^* for each $x \in \mathscr{L}$.
   - w^* is "independent" of any witness w w.r.t x.

Unique Witness Maps (UWM)



- $\mathsf{Setup}(\lambda)$: crs
- $\mathsf{map}(crs, x, w)$: w^*. The mapping is deterministic.
- $\mathsf{check}(crs, x, w^*)$: 1/0

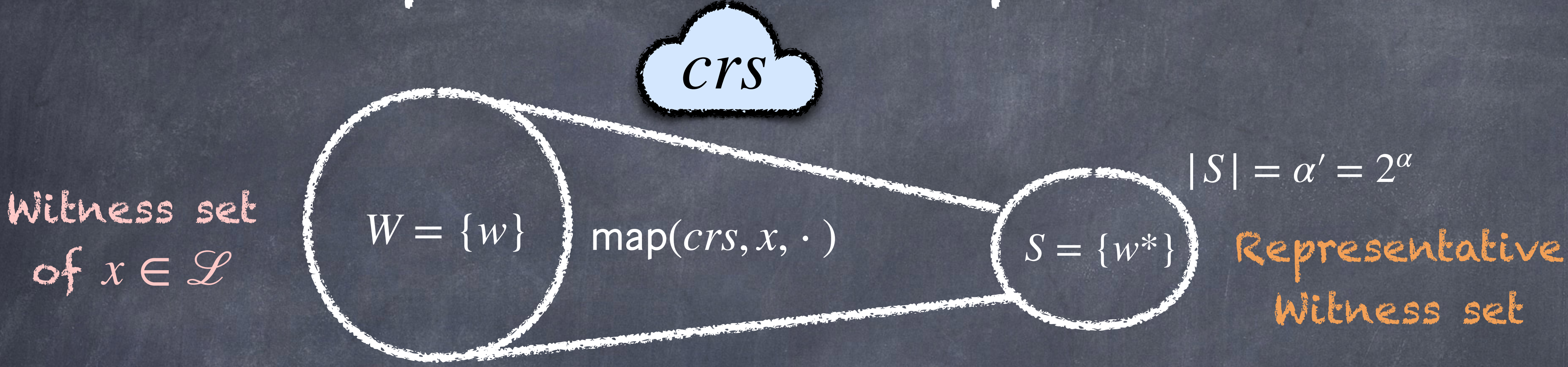
1. Completeness: If x is "true" ⇝ check o/p 1.
2. (Computational) Soundness: If x is "false" ⇝ check o/p 0.
3. Uniqueness: Unique proof w^* for each $x \in$

   – w^* is "independent" of any witness w w

NIWI with det. prover & verifier

Compact Witness Maps (CWM)

Compact Witness Maps (CWM)



crs

Witness set of $x \in \mathscr{L}$

$W = \{w\}$

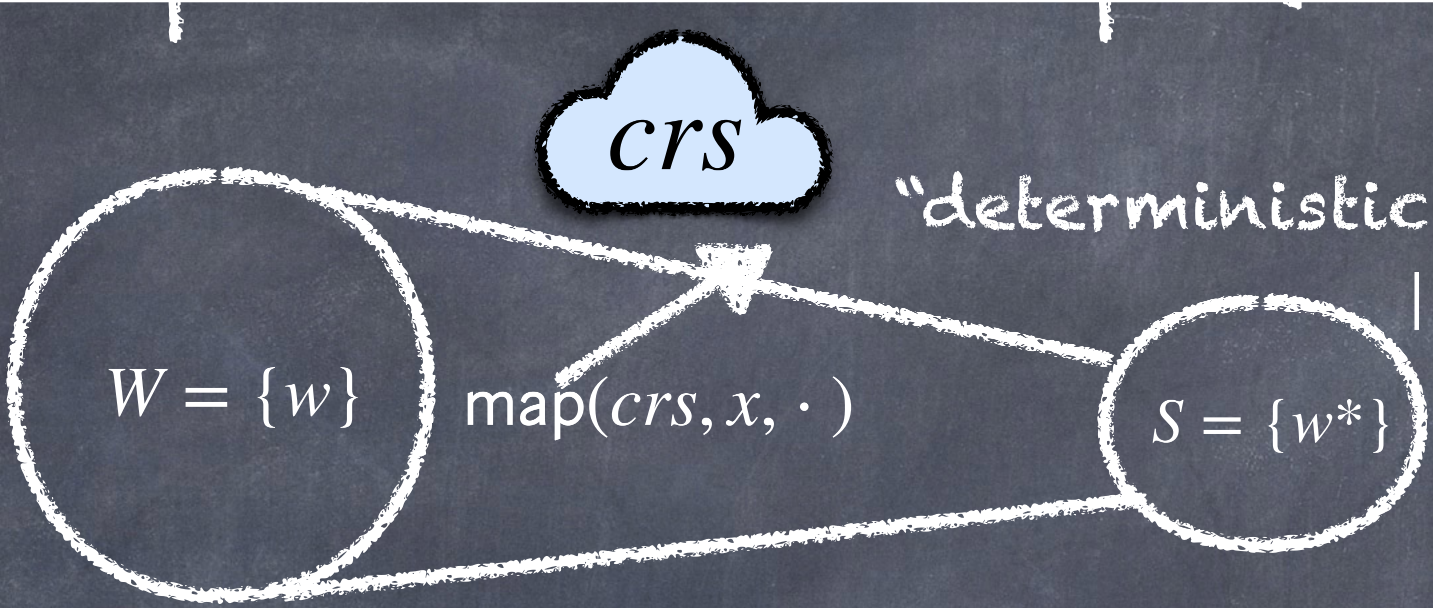
$\mathsf{map}(crs, x, \cdot\,)$

$S = \{w^*\}$

$|S| = \alpha' = 2^\alpha$

Representative Witness set

Compact Witness Maps (CWM)



Witness set of $x \in \mathscr{L}$

crs

"deterministic"

$|S| = \alpha' = 2^{\alpha}$

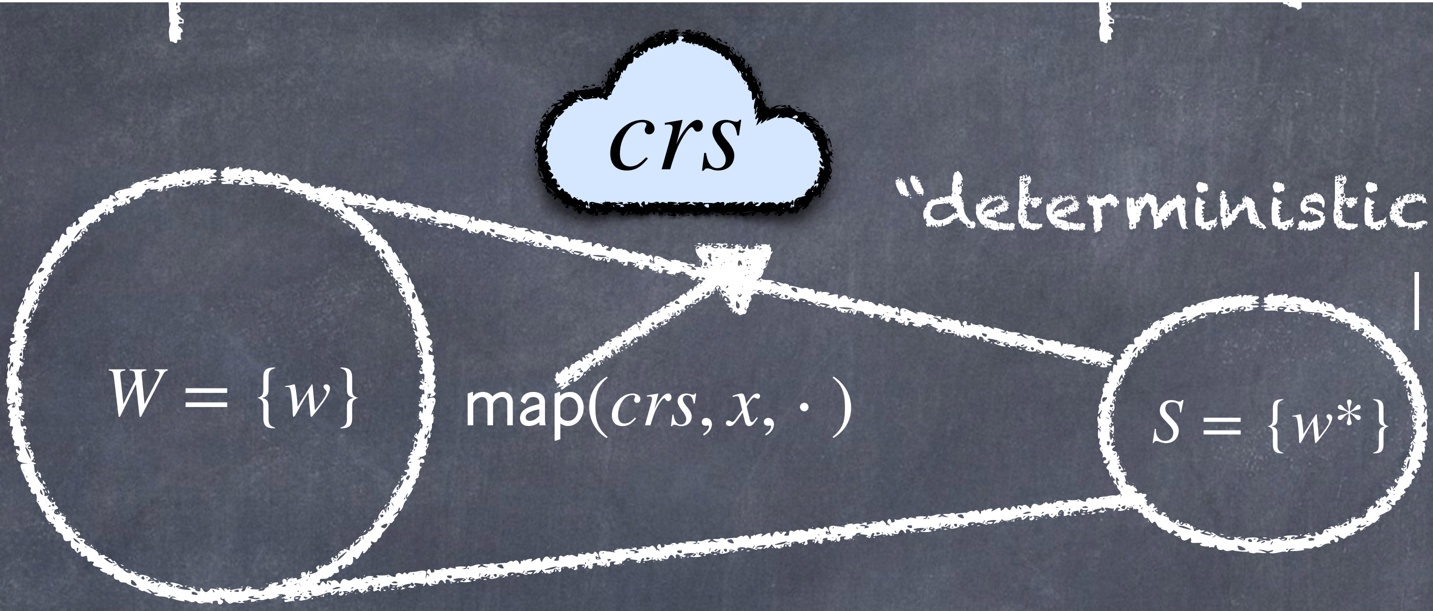
$W = \{w\}$

map$(crs, x, \cdot\,)$

$S = \{w^*\}$

Representative Witness set

Compact Witness Maps (CWM)



crs

"deterministic"

$|S| = \alpha' = 2^{\alpha}$

Witness set of $x \in \mathscr{L}$

$W = \{w\}$

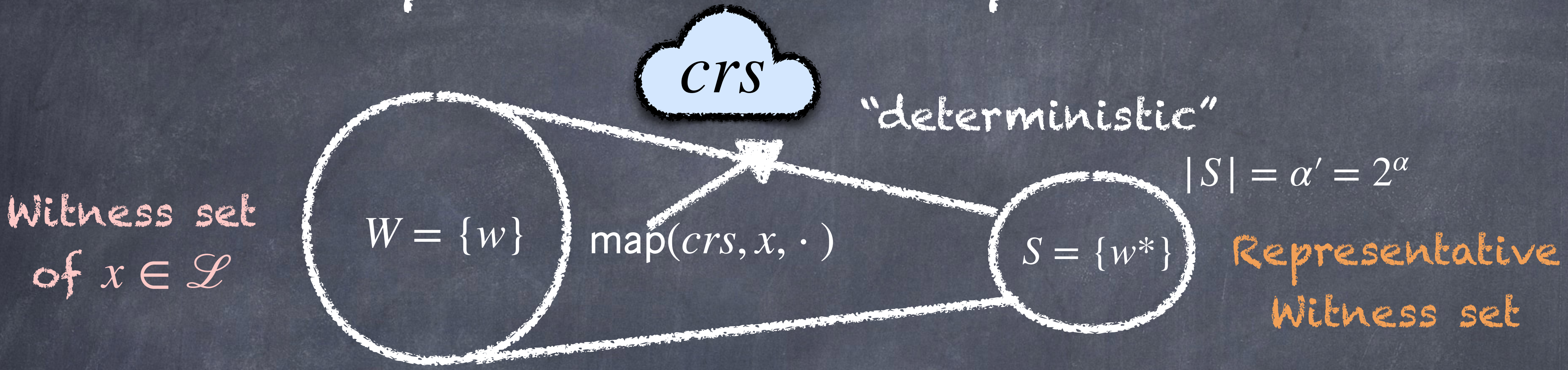
$\mathsf{map}(crs, x, \cdot\,)$

$S = \{w^*\}$

Representative Witness set

- setup, map, and check same as in CWM.

Compact Witness Maps (CWM)



crs

"deterministic"

$|S| = \alpha' = 2^\alpha$

Witness set of $x \in \mathscr{L}$

$W = \{w\}$

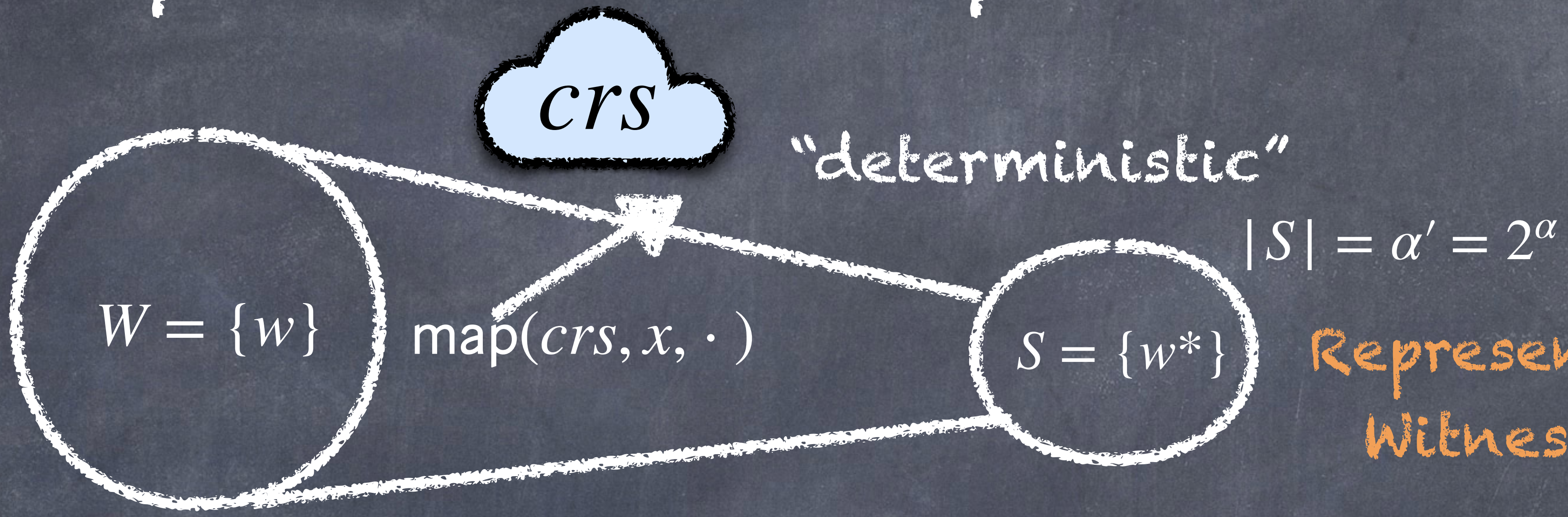
$\mathsf{map}(crs, x, \cdot\,)$

$S = \{w*\}$

Representative Witness set

- setup, map, and check same as in CWM.
- Want completeness and (comp) soundness to hold as in CWM.

Compact Witness Maps (CWM)

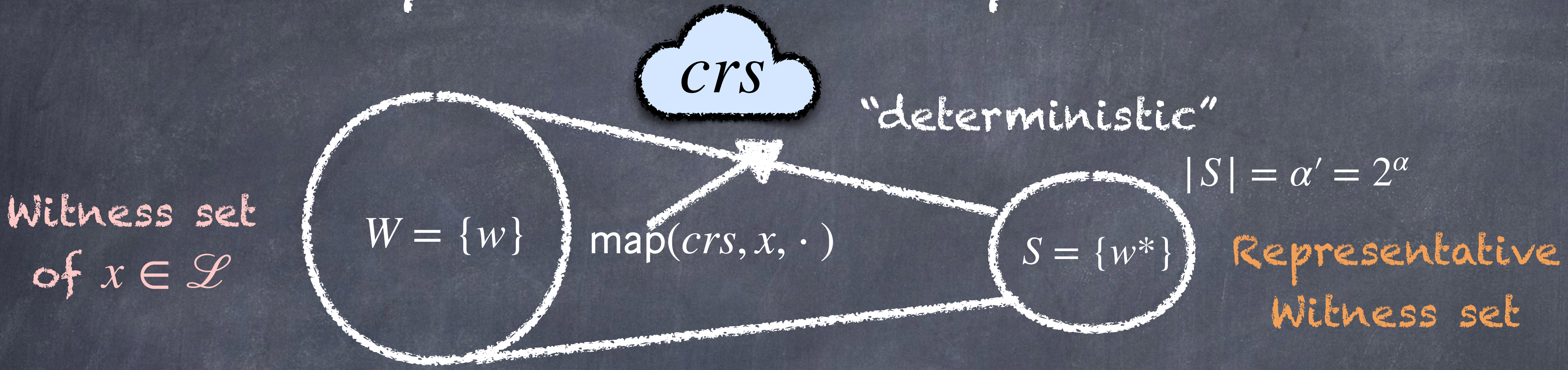


Witness set of $x \in \mathscr{L}$

Representative Witness set

- setup, map, and check same as in CWM.
- Want completeness and (comp) soundness to hold as in CWM.
- α-Lossiness: Let $S = \{w^* = \text{map}(crs, x, w)\}$. We require that $|S| = 2^\alpha << |W|$.

Compact Witness Maps (CWM)



- setup, map, and check same as in CWM.
- Want completeness and (comp) soundness to hold as in CWM.
- α-Lossiness: Let $S = \{w^* = \text{map}(crs, x, w)\}$. We require that $|S| = 2^{\alpha} << |W|$.
- UWM is α-lossy CWM with $\alpha = 0$.

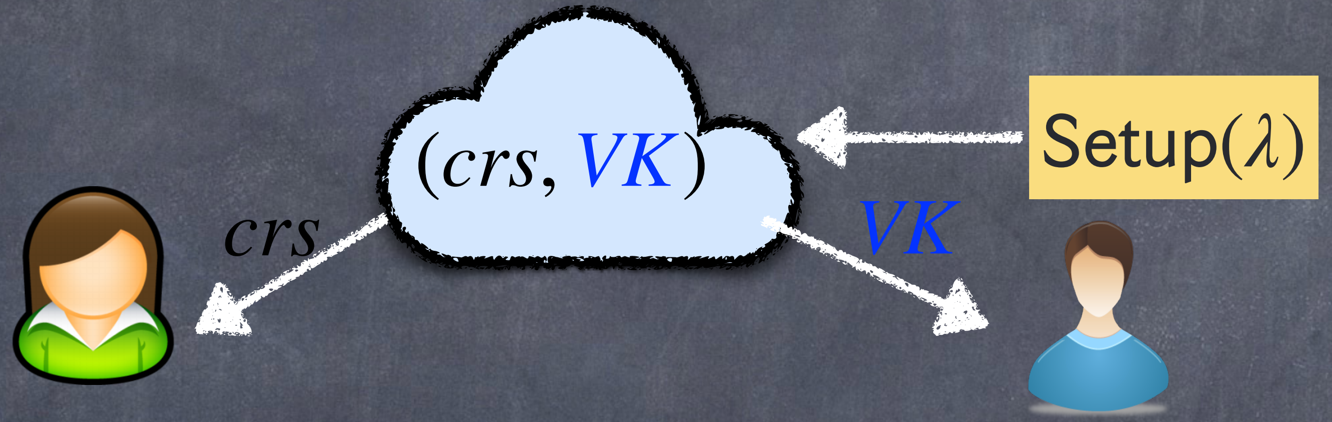
Designated-Verifier Unique Witness Maps (DV-UWM)

Designated-Verifier Unique Witness Maps (DV-UWM)

(crs, VK)

$\mathsf{Setup}(\lambda)$

Designated-Verifier Unique Witness Maps (DV-UWM)



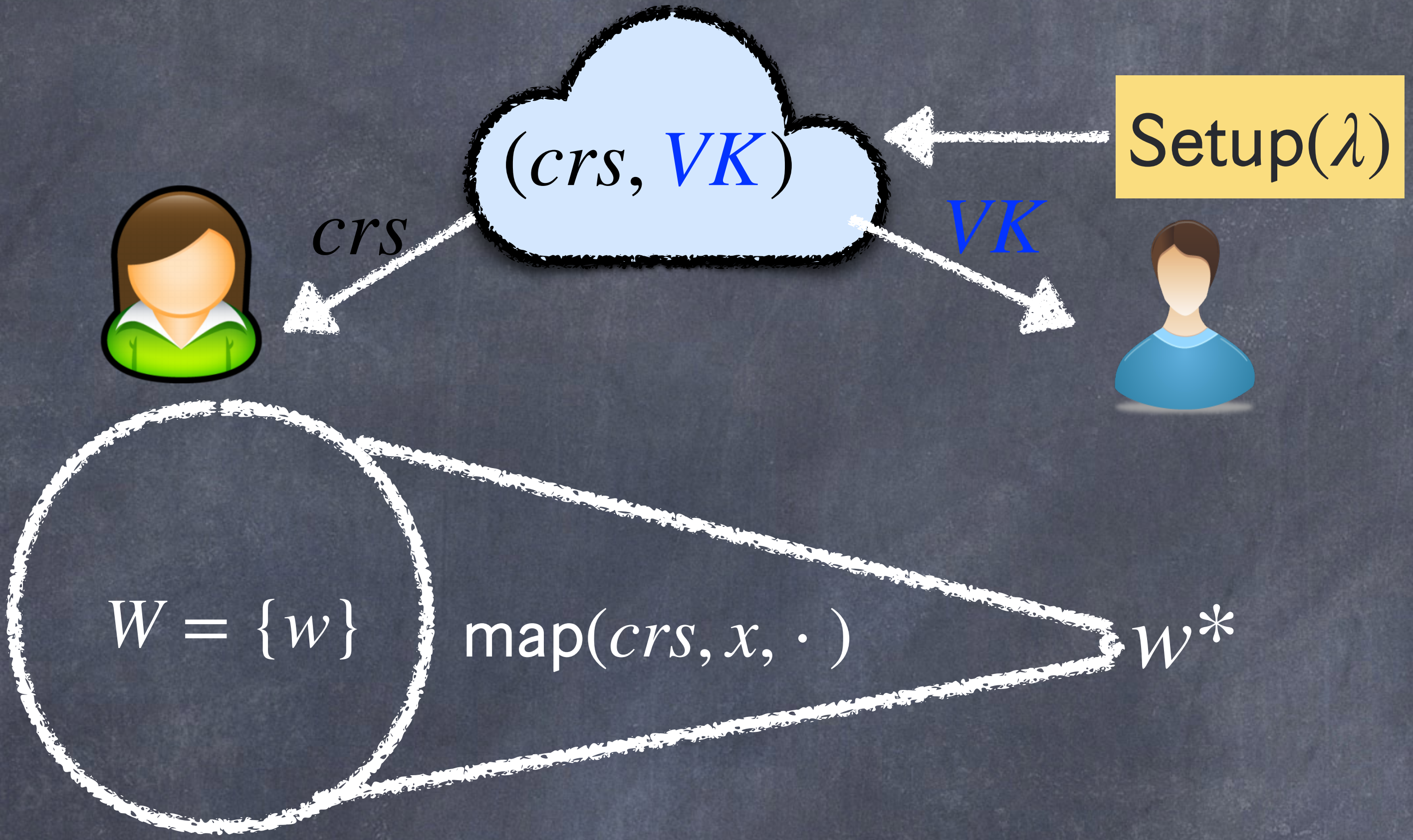
(crs, VK)

Setup(λ)

crs

VK

Designated-Verifier Unique Witness Maps (DV-UWM)



(crs, VK)

Setup(λ)

crs

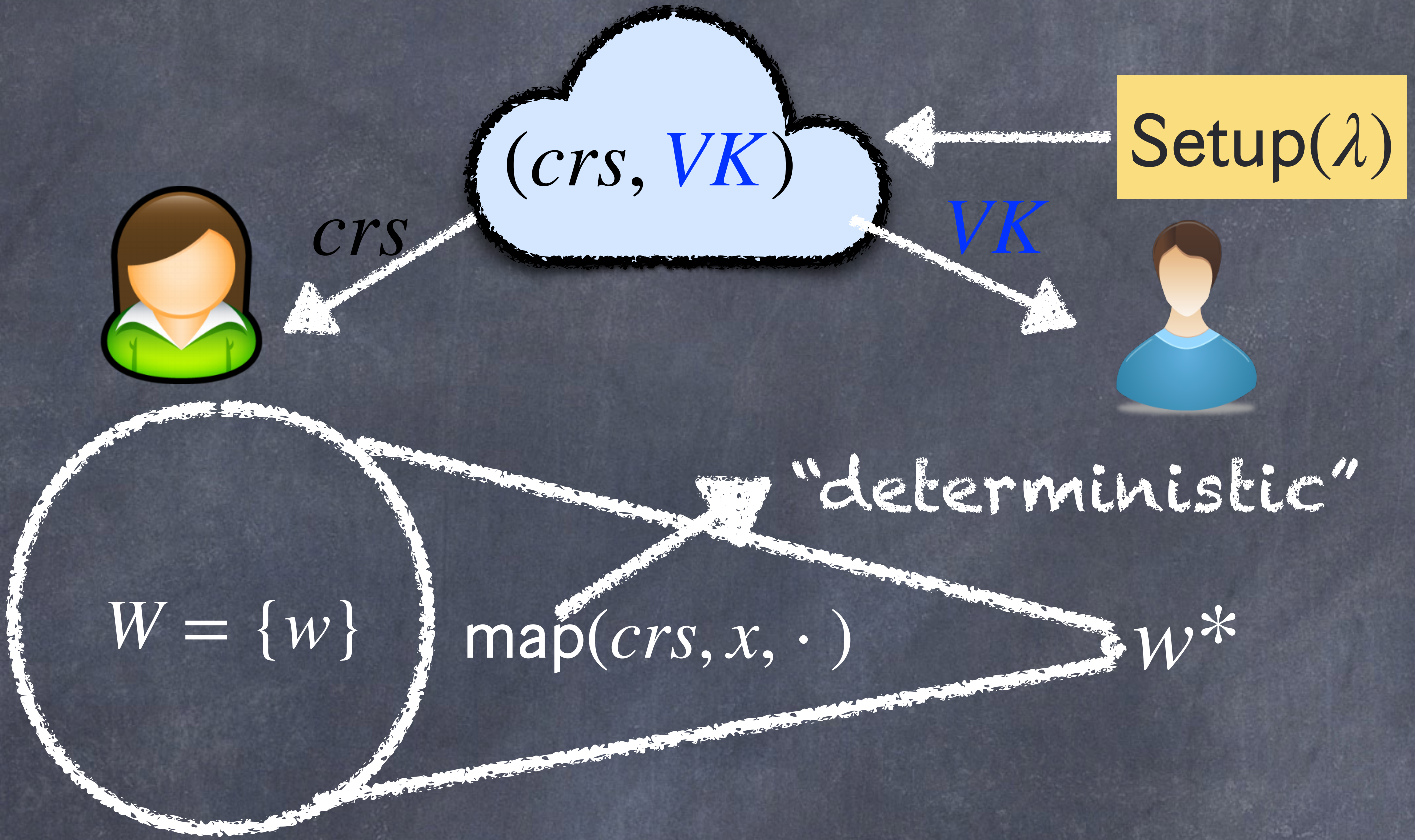
VK

$W = \{w\}$

map($crs, x, \cdot$)

w^*

Designated-Verifier Unique Witness Maps (DV-UWM)



(crs, VK)

Setup(λ)

crs

VK

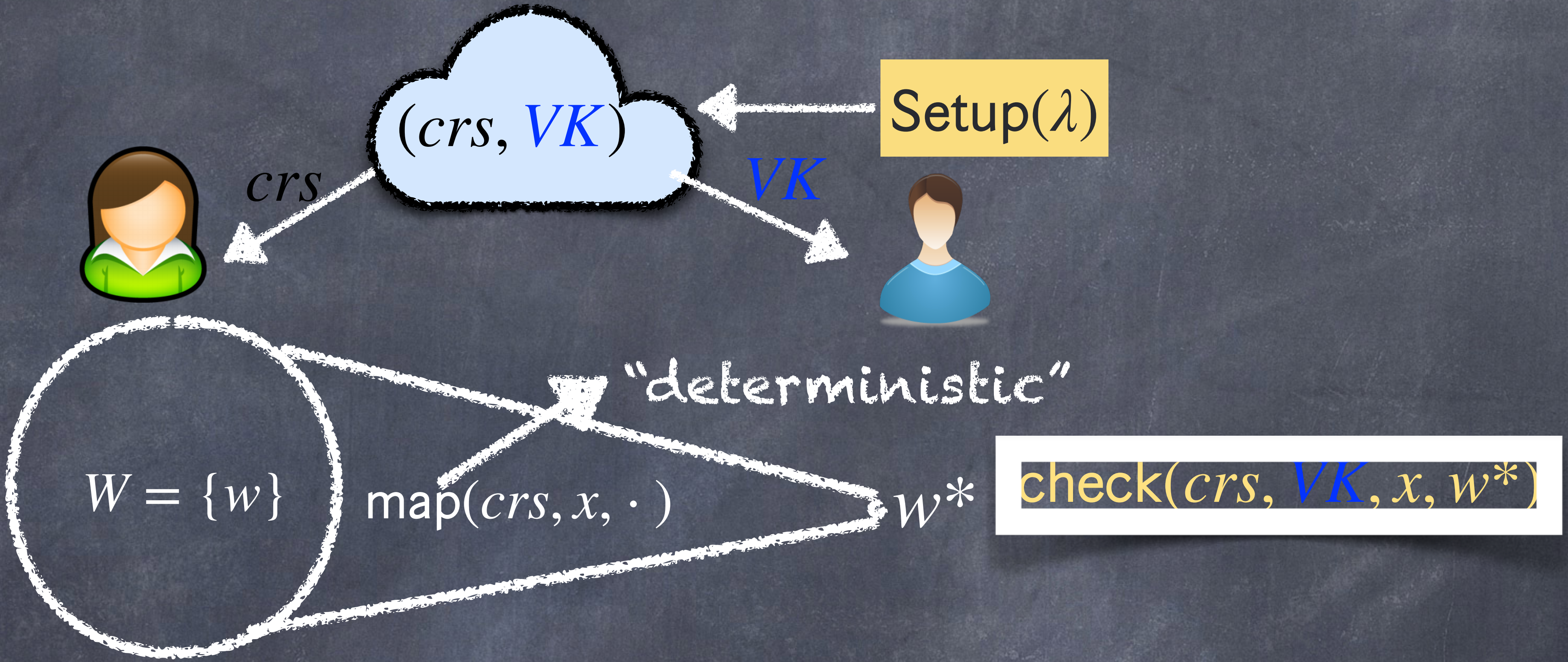
"deterministic"

$W = \{w\}$

map$(crs, x, \cdot\,)$

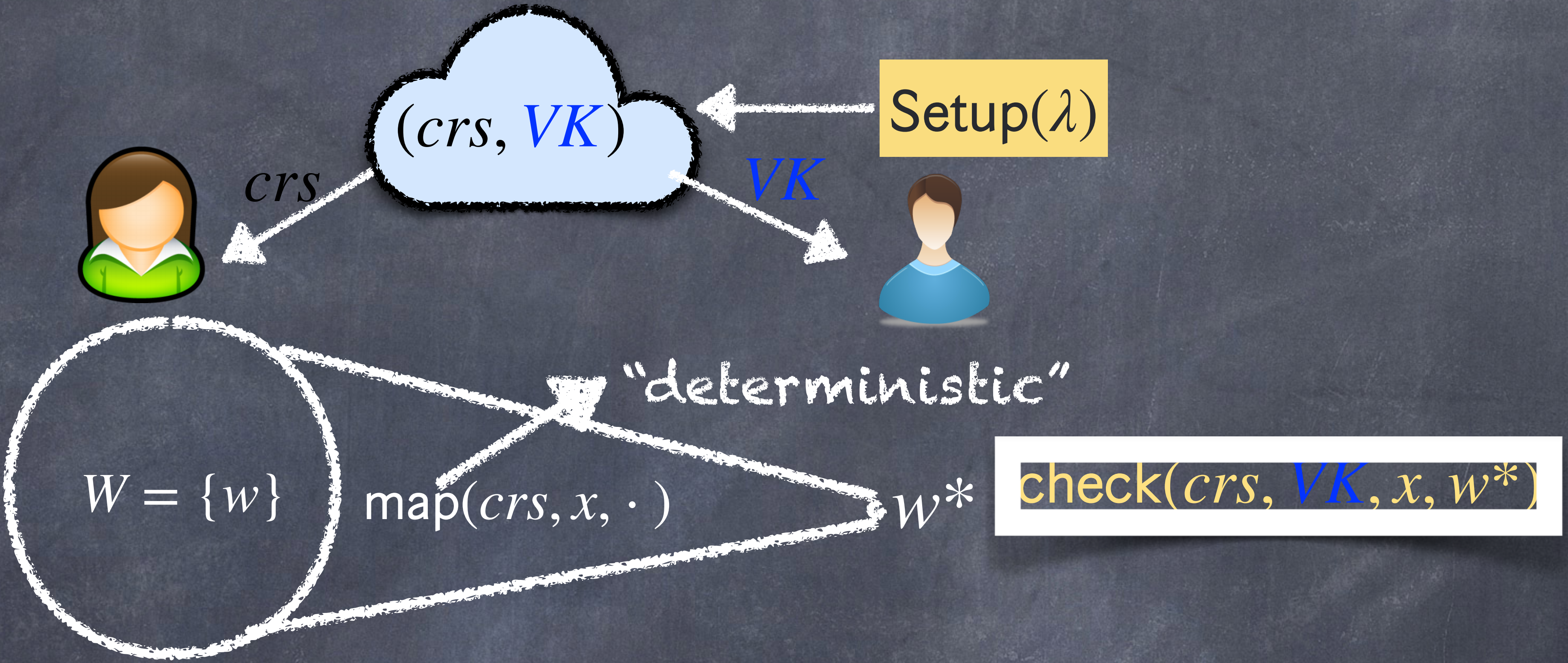
w^*

Designated-Verifier Unique Witness Maps (DV-UWM)



Designated-Verifier Unique Witness Maps (DV-UWM)

Witness set
of $x \in \mathscr{L}$



(crs, VK)

$\mathsf{Setup}(\lambda)$

crs

VK

"deterministic"

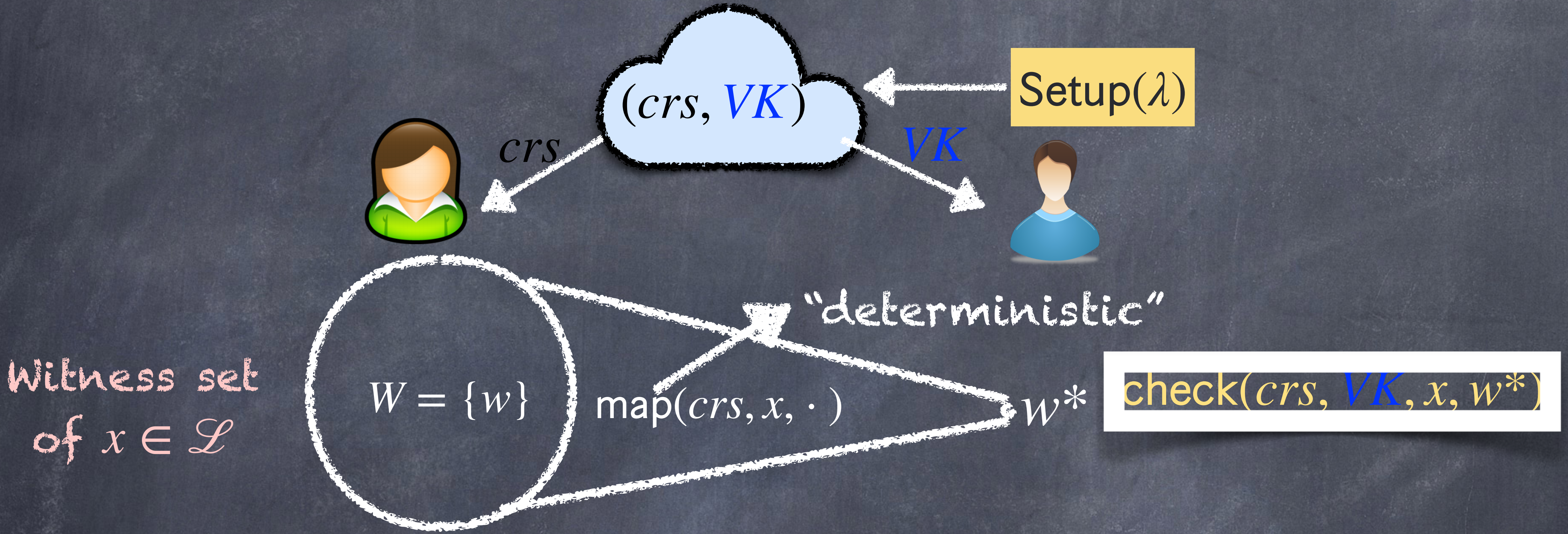
$W = \{w\}$

$\mathsf{map}(crs, x, \cdot\,)$

w^*

$\mathsf{check}(crs, VK, x, w^*)$

Designated-Verifier Unique Witness Maps (DV-UWM)



(crs, VK)

Setup(λ)

crs

VK

Witness set of $x \in \mathscr{L}$

$W = \{w\}$

map$(crs, x, \cdot\,)$

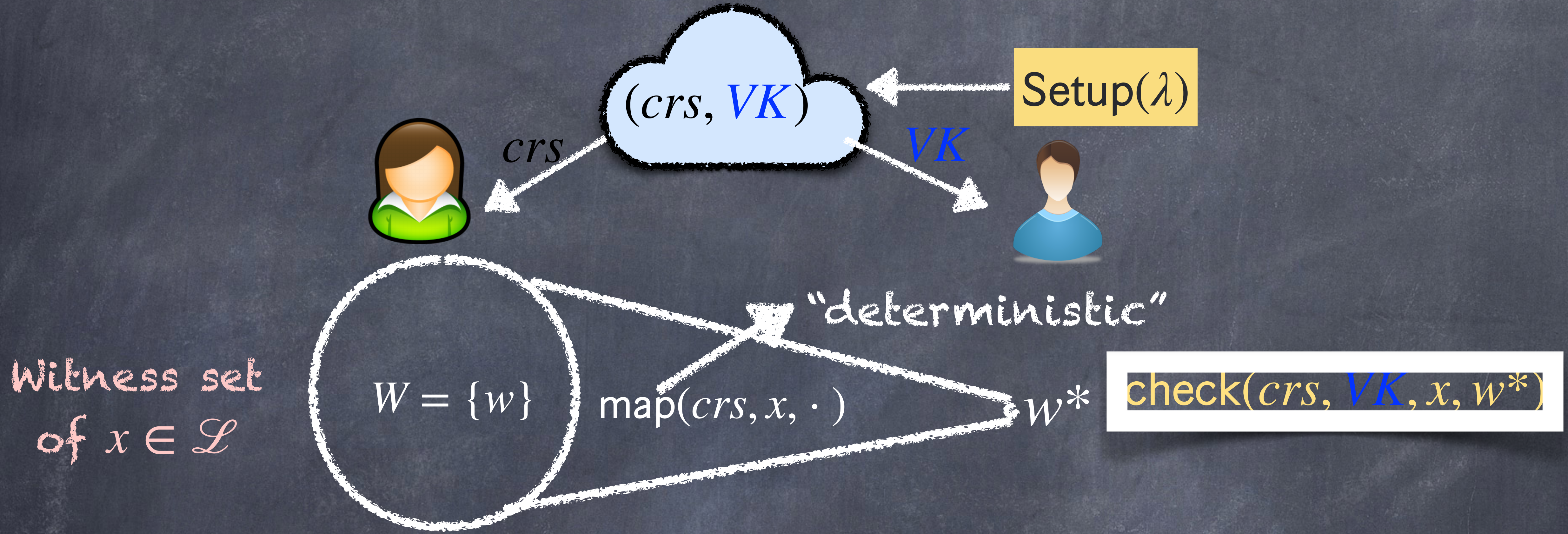
"deterministic"

$w*$

check$(crs, VK, x, w*)$

1. **Completeness:** If x is "true" $\rightsquigarrow$ check o/p 1.

Designated-Verifier Unique Witness Maps (DV-UWM)



(crs, VK)

Setup(λ)

crs

VK

"deterministic"

Witness set of $x \in \mathscr{L}$

$W = \{w\}$

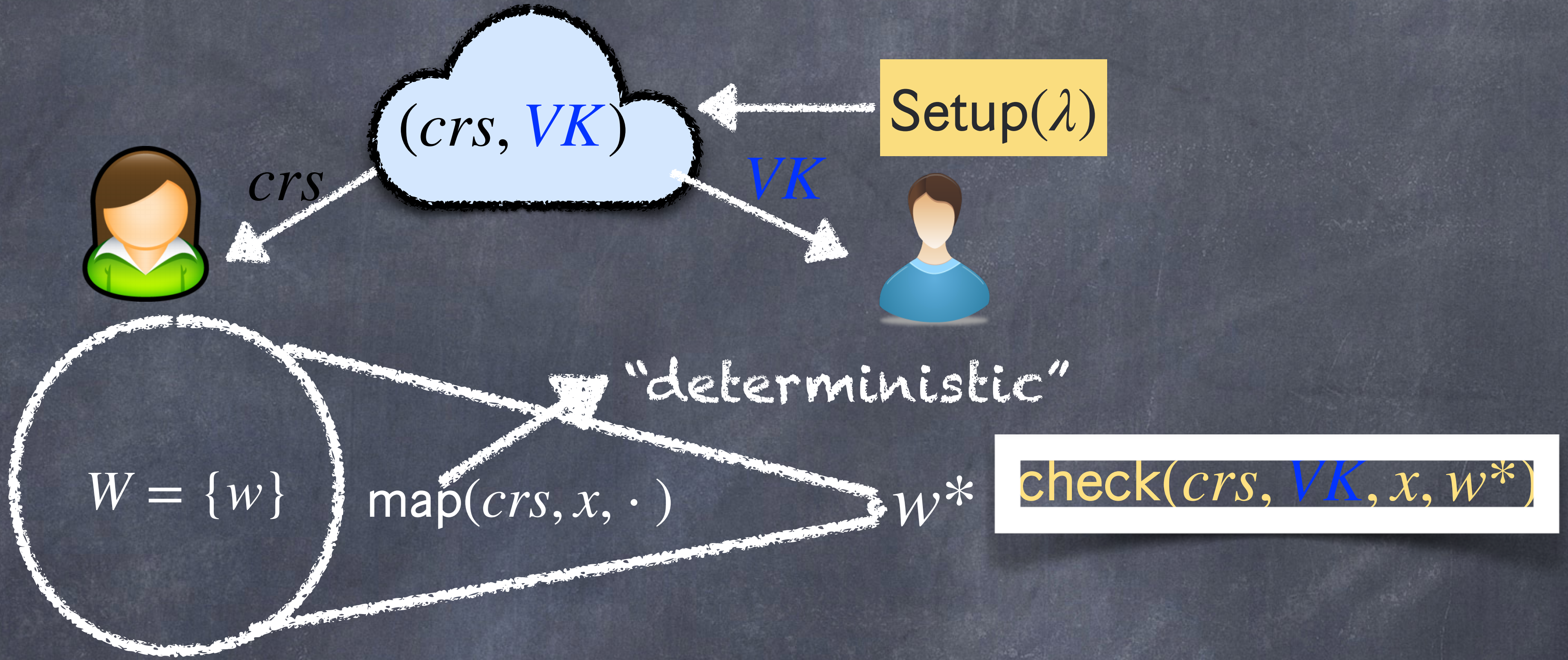
map$(crs, x, \cdot\,)$

w^*

check(crs, VK, x, w^*)

1. **Completeness:** If x is "true" ⤳ check o/p 1.
2. **(Non-Reusable) Soundness::** If x is "false" ⤳ check o/p 0.

Designated-Verifier Unique Witness Maps (DV-UWM)



(crs, VK)

Setup(λ)

crs

VK

"deterministic"

Witness set of $x \in \mathscr{L}$

$W = \{w\}$

map$(crs, x, \cdot)$

w^*

check(crs, VK, x, w^*)

1. **Completeness:** If x is "true" ⇝ check o/p 1.
2. **(Non-Reusable) Soundness::** If x is "false" ⇝ check o/p 0.
3. **(Reusable) Soundness:** If x is "false" ⇝ check o/p 0, even if the adversary gets oracle access to check$(VK, \cdot, \cdot)$

Prior Results on witness maps [CPW'20]

Prior Results on witness maps [CPW'20]

- UWM (along with LTDFs) implies (bounded) leakage and tamper-resilient signatures.

Prior Results on witness maps [CPW'20]

- UWM (along with LTDFs) implies (bounded) leakage and tamper-resilient signatures.
- Indistinguishability obfuscation (iO) + OWF => UWM

Prior Results on witness maps [CPW'20]

- UWM (along with LTDFs) implies (bounded) leakage and tamper-resilient signatures.
- Indistinguishability obfuscation (iO) + OWF => UWM
- UWM => Witness encryption (WE)

Prior Results on witness maps [CPW'20]

- UWM (along with LTDFs) implies (bounded) leakage and tamper-resilient signatures.
- Indistinguishability obfuscation (iO) + OWF => UWM
- UWM => Witness encryption (WE)
  - In fact α-CWM for $\alpha = O(\log \lambda)$ implies WE.

Our Contributions

Our Contributions

Our Contributions

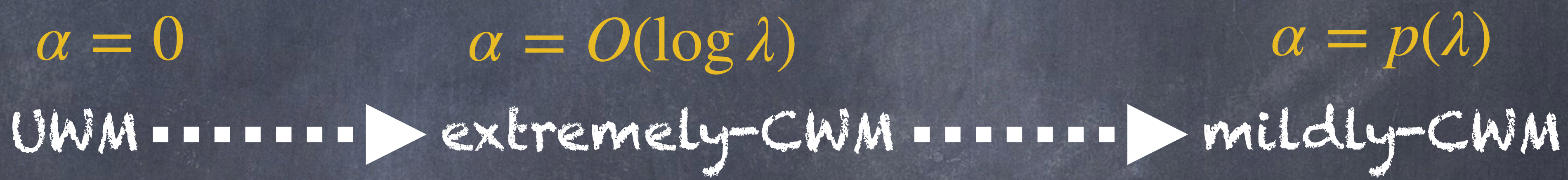
- Undertake a thorough study of witness maps and connect them to other crypto primitives.

Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"

Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
    - Contribute to the study of "functional compression"

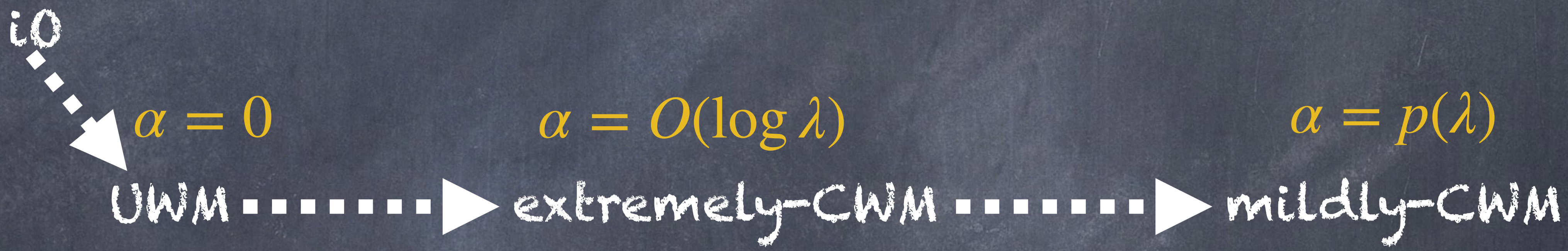


$\alpha = 0$ $\alpha = O(\log \lambda)$ $\alpha = p(\lambda)$

UWM ⋯⋯▶ extremely-CWM ⋯⋯▶ mildly-CWM

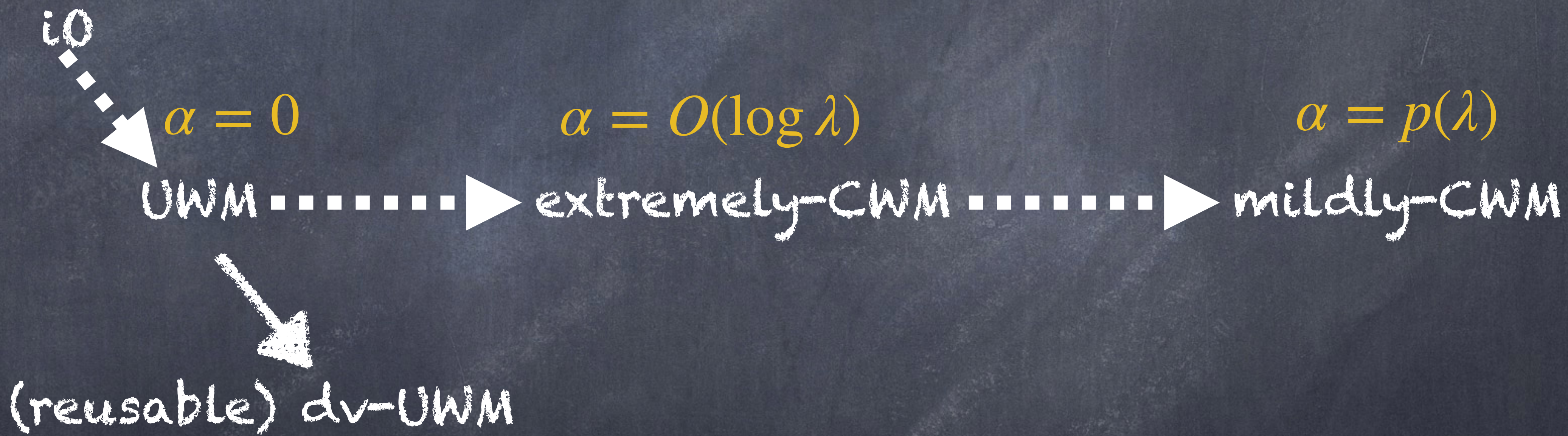
Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



iO

$\alpha = 0$

$\alpha = O(\log \lambda)$

$\alpha = p(\lambda)$

UWM

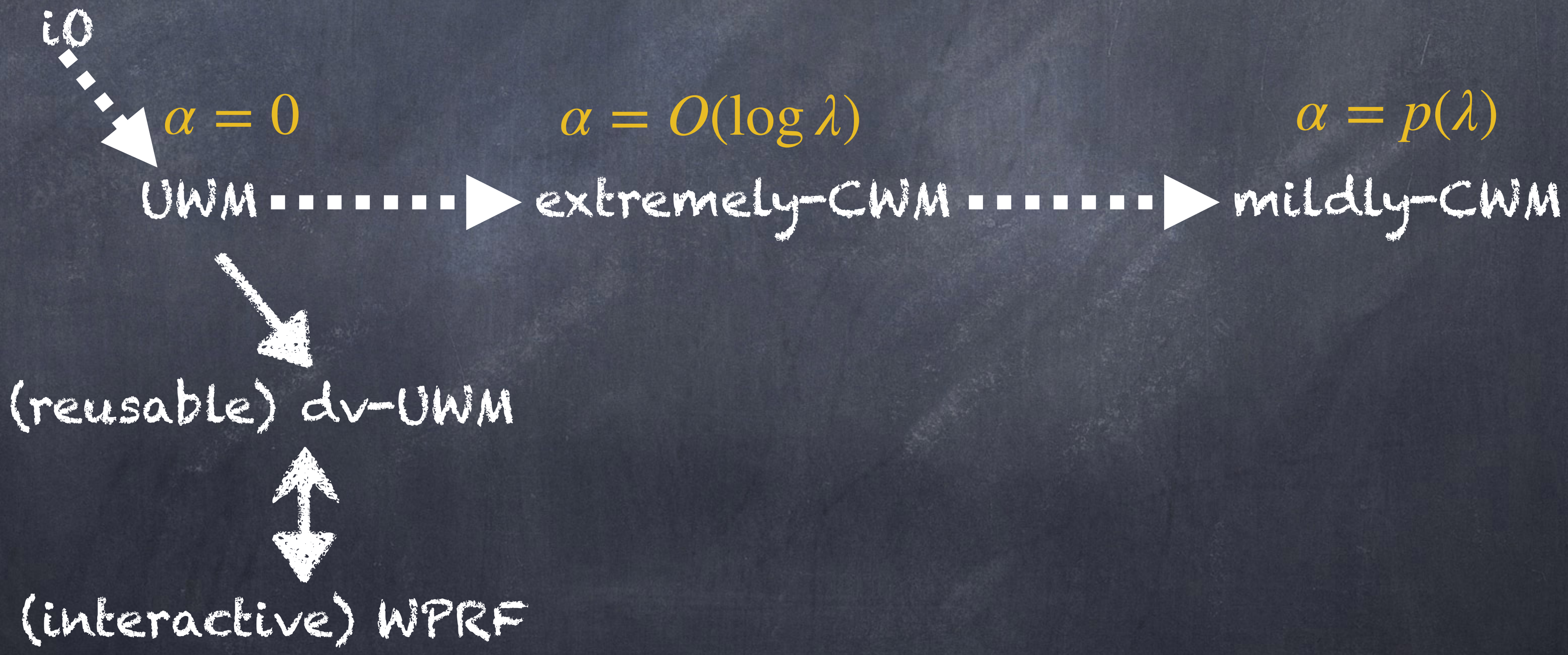
extremely-CWM

mildly-CWM

(reusable) dv-UWM

Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



iO

$\alpha = 0$

$\alpha = O(\log \lambda)$

$\alpha = p(\lambda)$

UWM

extremely-CWM

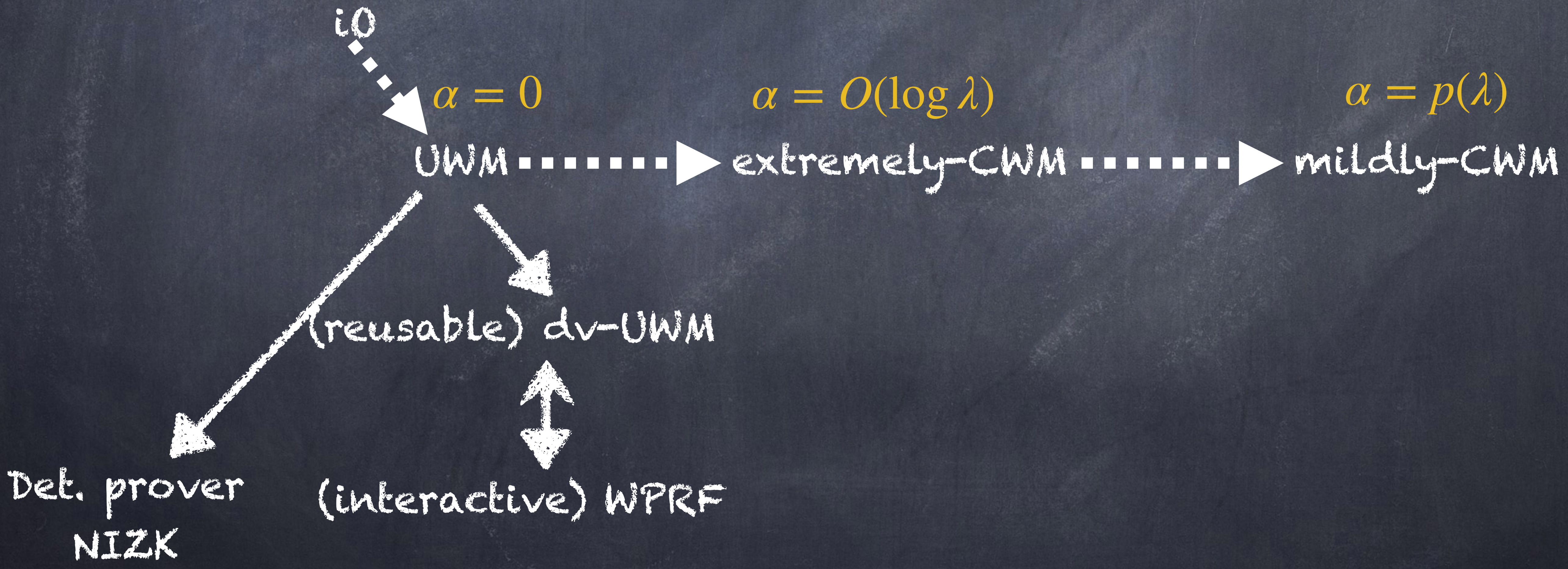
mildly-CWM

(reusable) dv-UWM

(interactive) WPRF

Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



iO

$\alpha = 0$

$\alpha = O(\log \lambda)$

$\alpha = p(\lambda)$

UWM

extremely-CWM

mildly-CWM

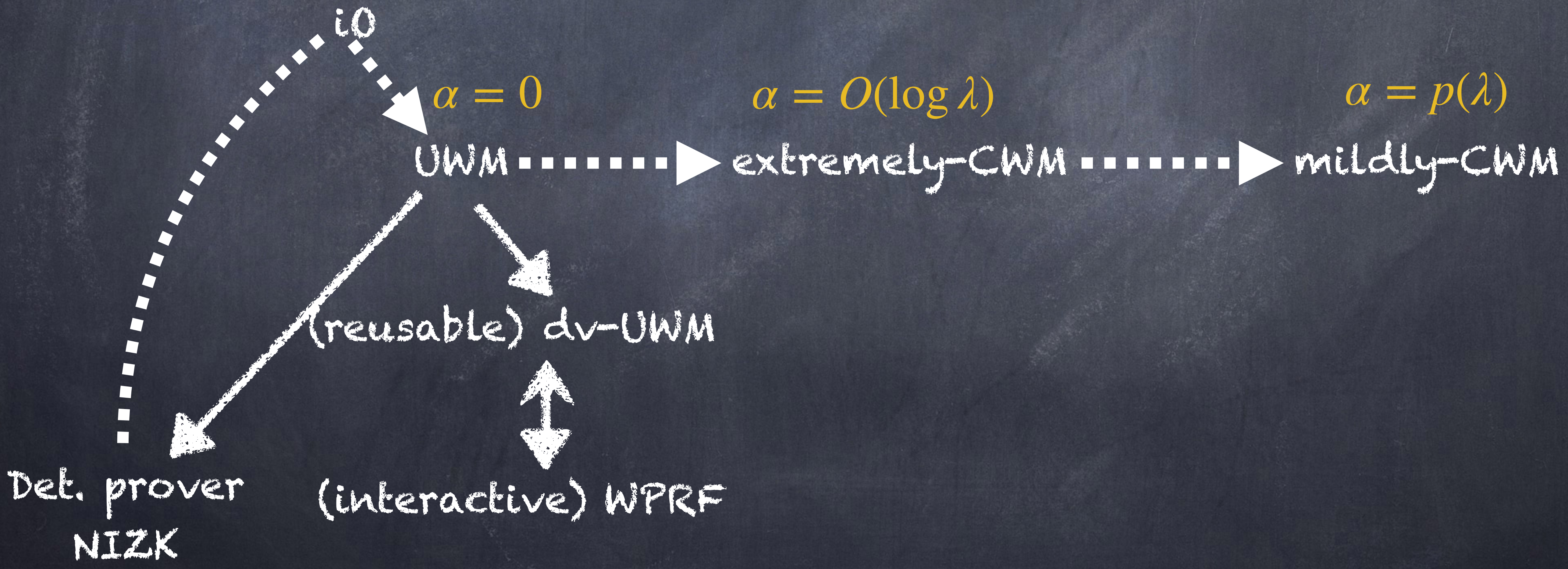
(reusable) dv-UWM

Det. prover NIZK

(interactive) WPRF

Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



iO

$\alpha = 0$

$\alpha = O(\log \lambda)$

$\alpha = p(\lambda)$

UWM

extremely-CWM

mildly-CWM

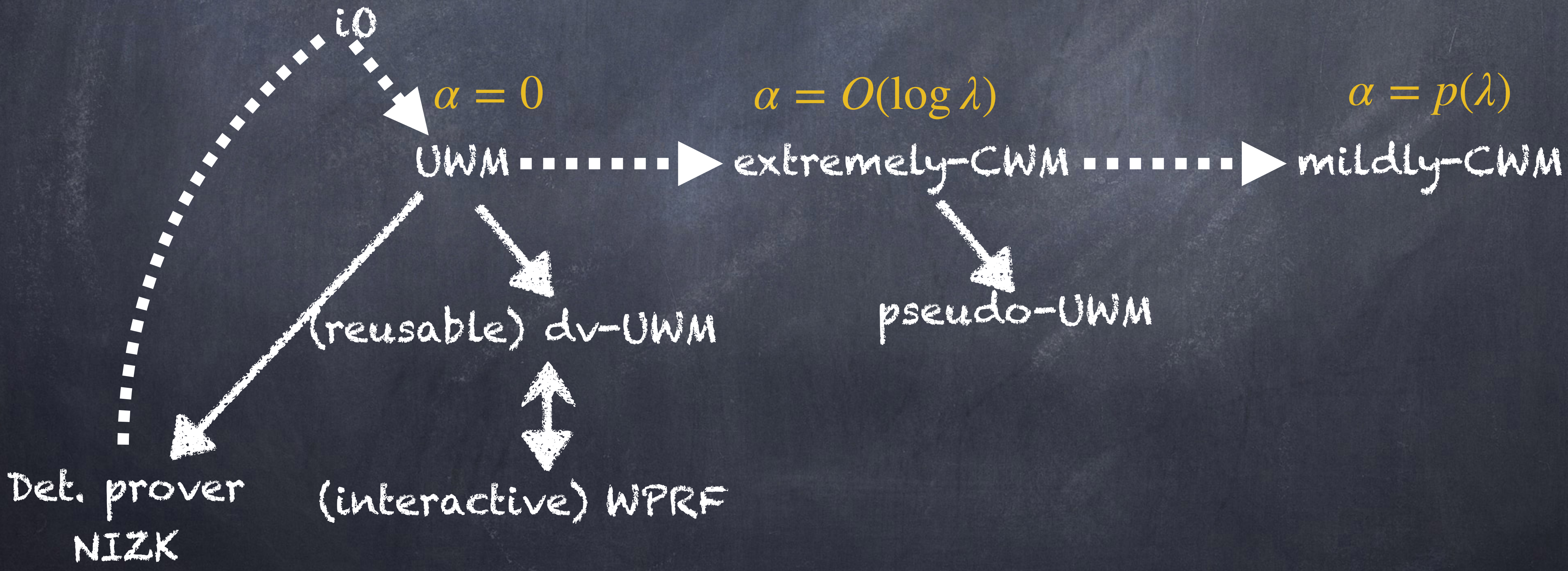
(reusable) dv-UWM

Det. prover NIZK

(interactive) WPRF

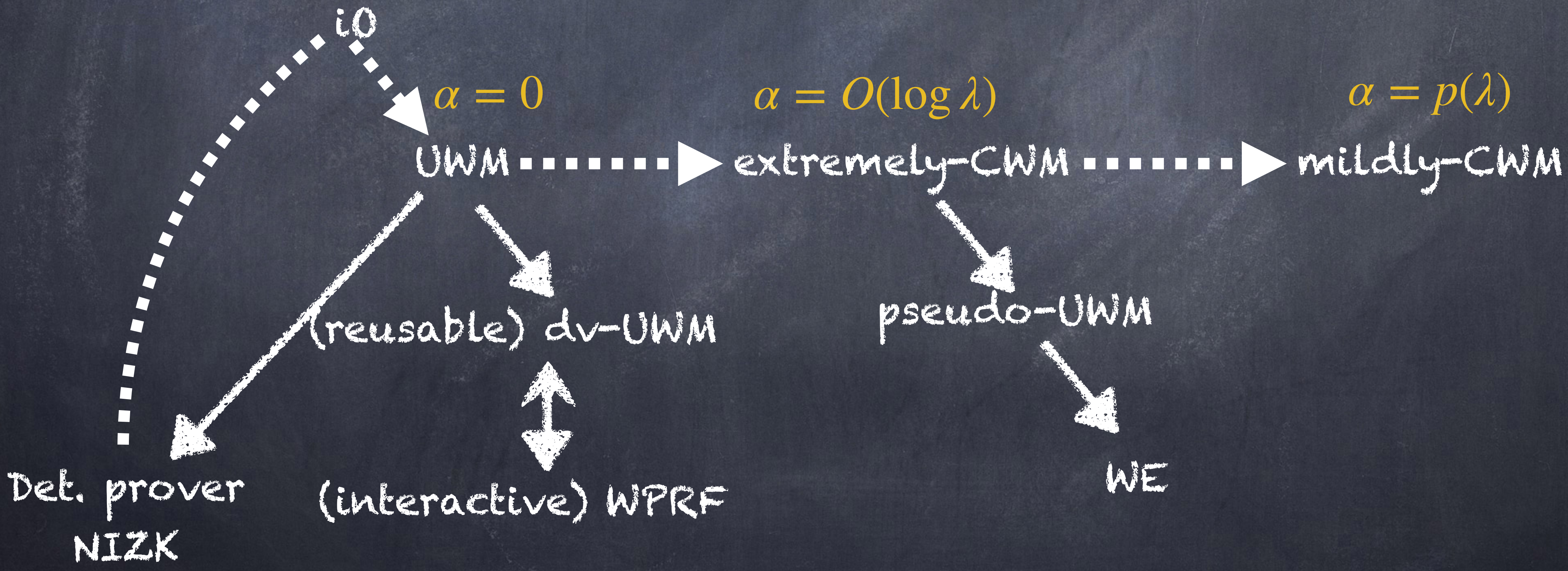
Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



iO

$\alpha = 0$

UWM

$\alpha = O(\log \lambda)$

extremely-CWM

$\alpha = p(\lambda)$

mildly-CWM

(reusable) dv-UWM

pseudo-UWM

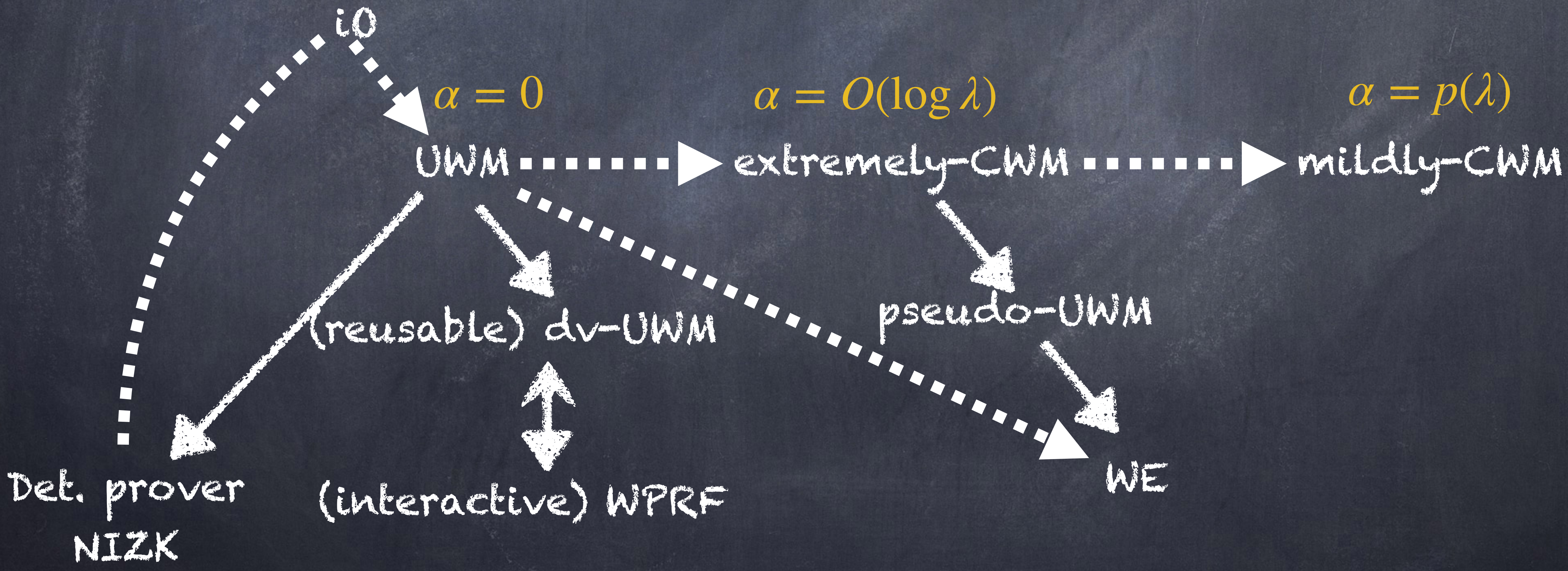
Det. prover NIZK

(interactive) WPRF

WE

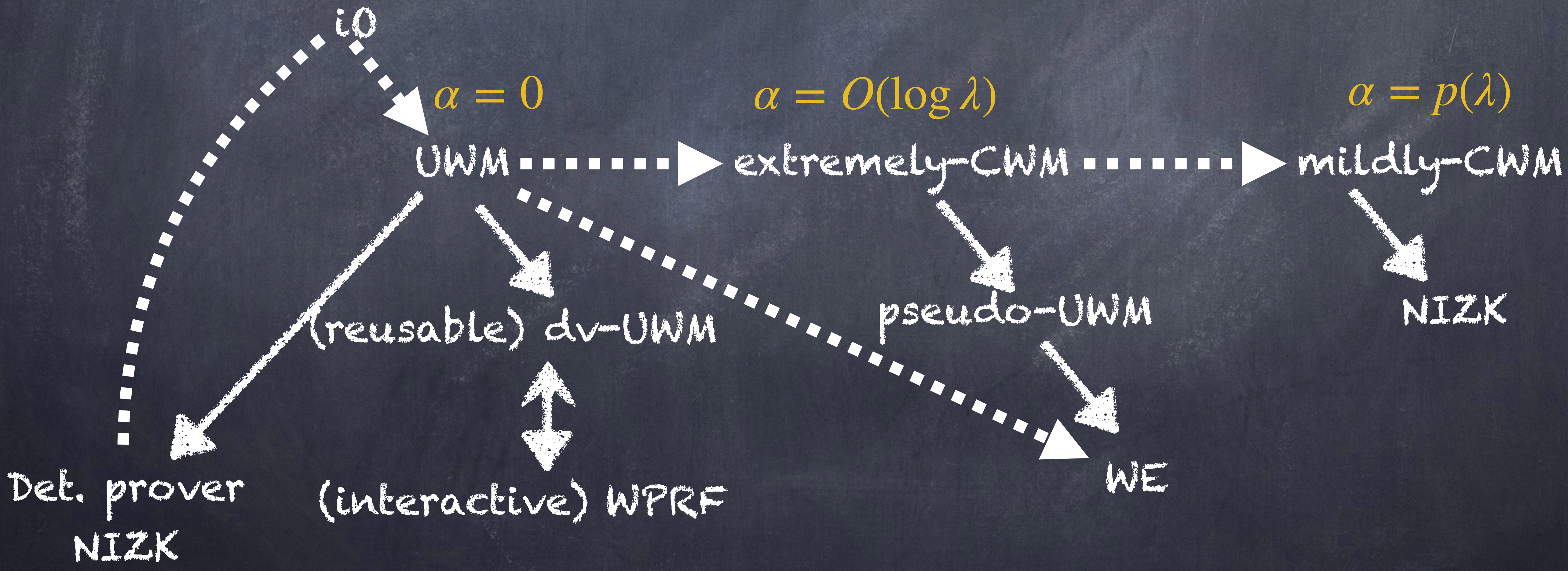
Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



iO

$\alpha = 0$

UWM

$\alpha = O(\log \lambda)$

extremely-CWM

$\alpha = p(\lambda)$

mildly-CWM

(reusable) dv-UWM

pseudo-UWM

NIZK

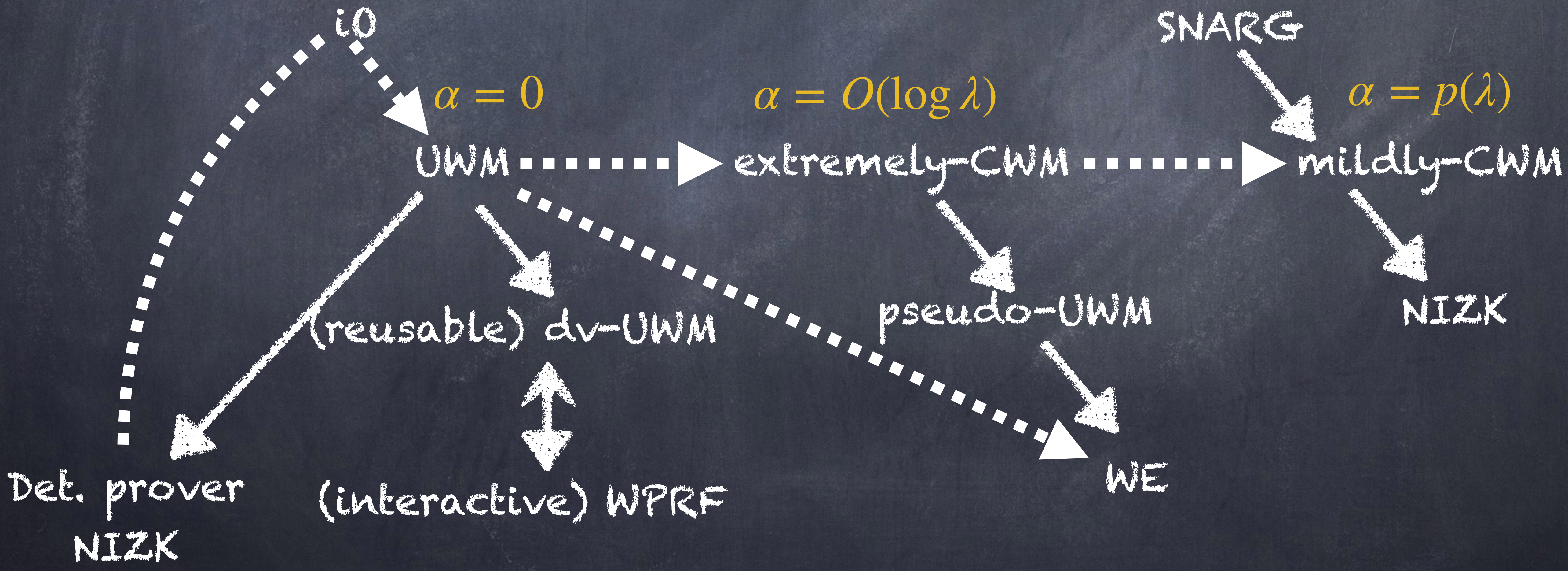
Det. prover NIZK

(interactive) WPRF

WE

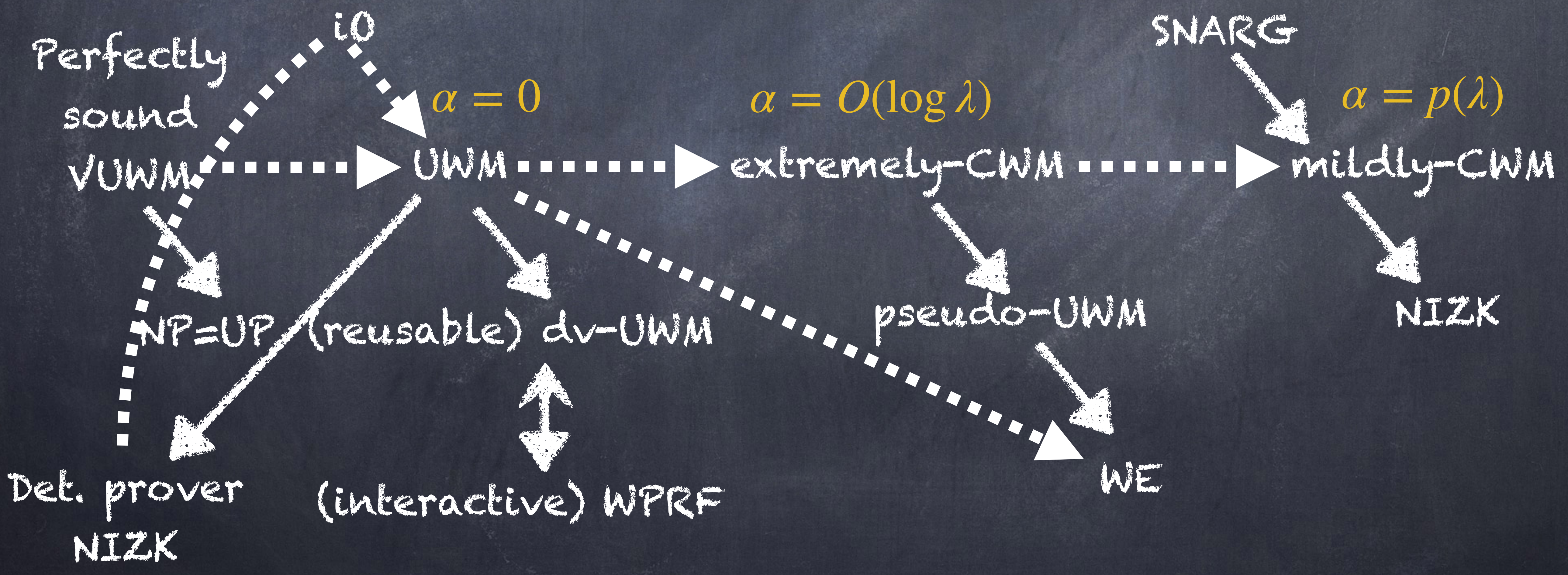
Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



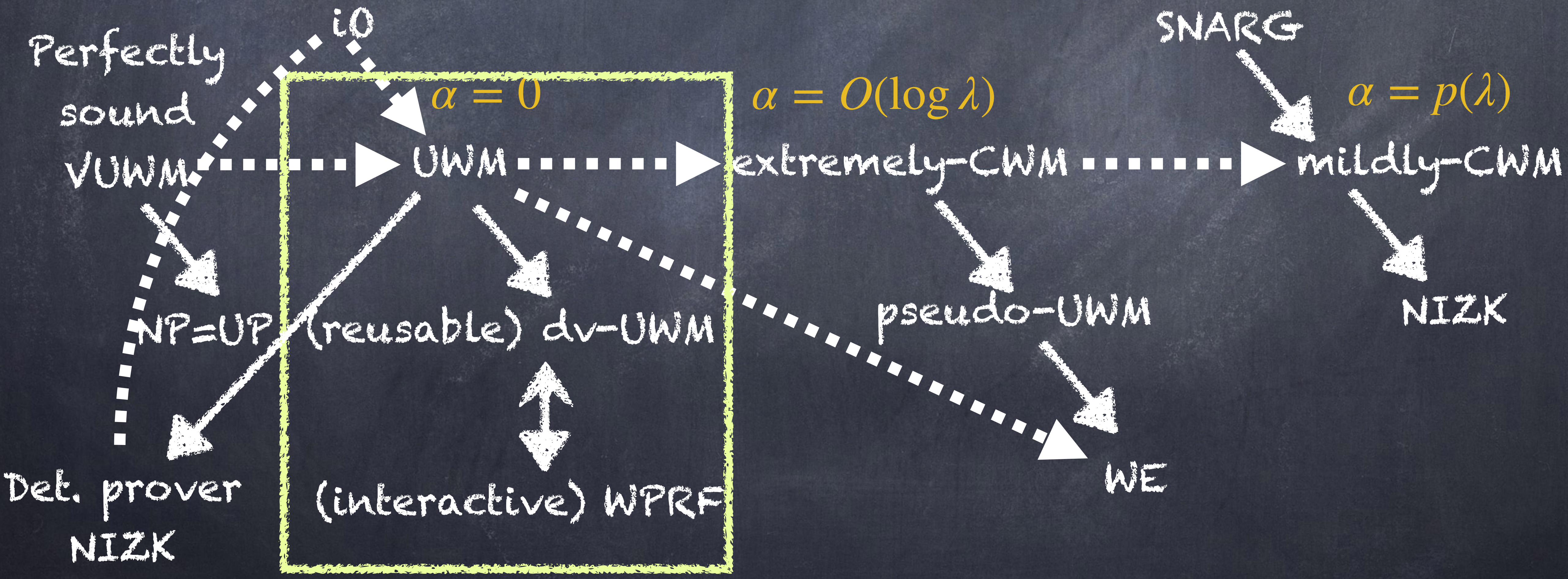
Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
  - Contribute to the study of "functional compression"



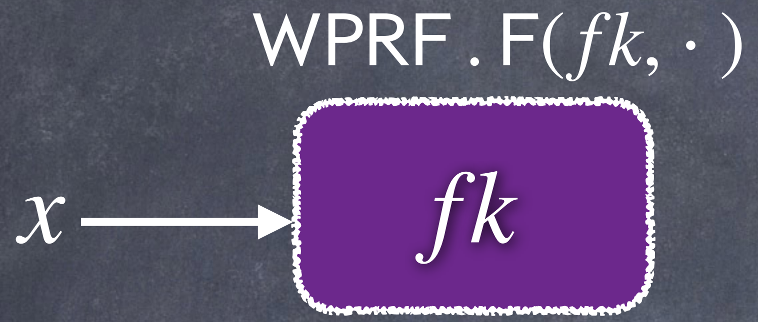
Our Contributions

- Undertake a thorough study of witness maps and connect them to other crypto primitives.
    - Contribute to the study of "functional compression"



Equivalence of dv-UWM and Witness PRF

Equivalence of dv-UWM and Witness PRF

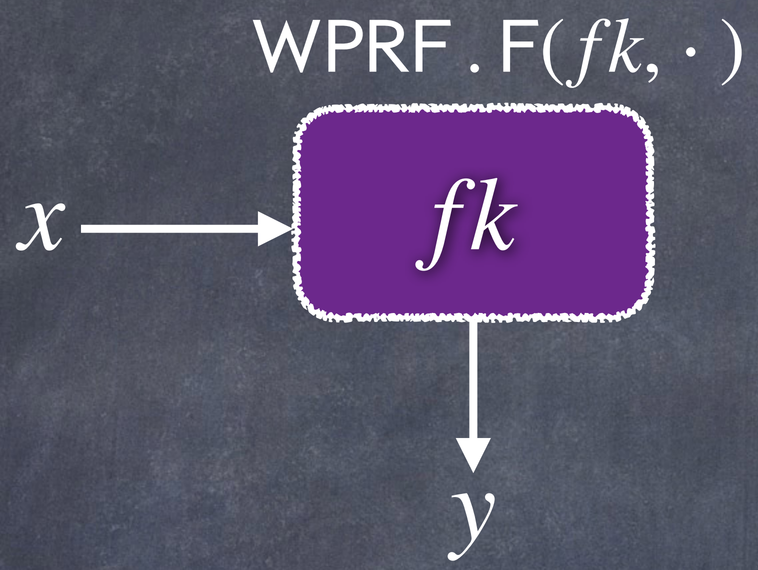


$\mathsf{WPRF}.\mathsf{F}(fk, \cdot)$

x

fk

Equivalence of dv-UWM and Witness PRF



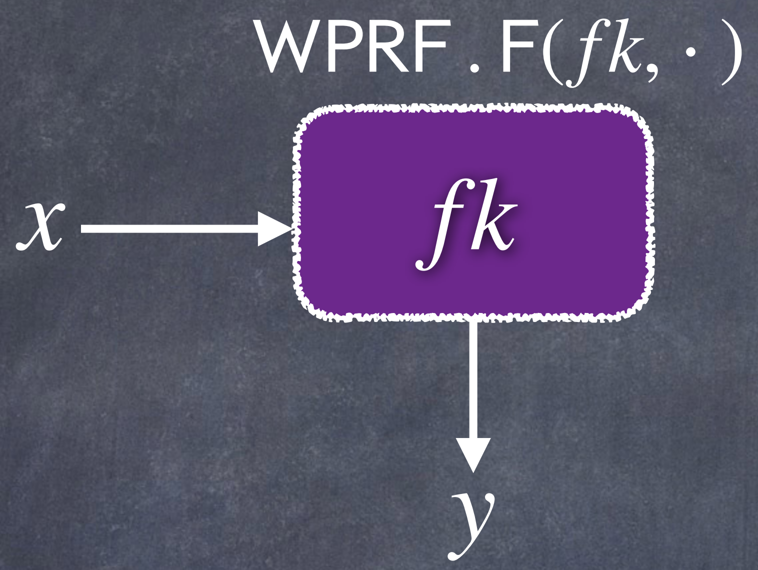
$\mathsf{WPRF.F}(fk, \cdot)$

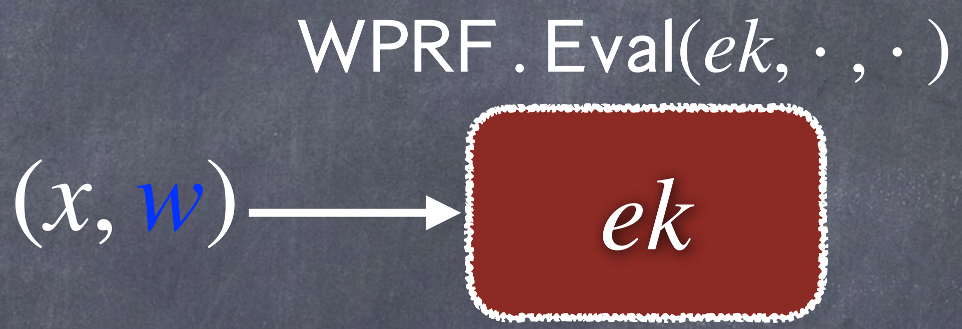
fk

x

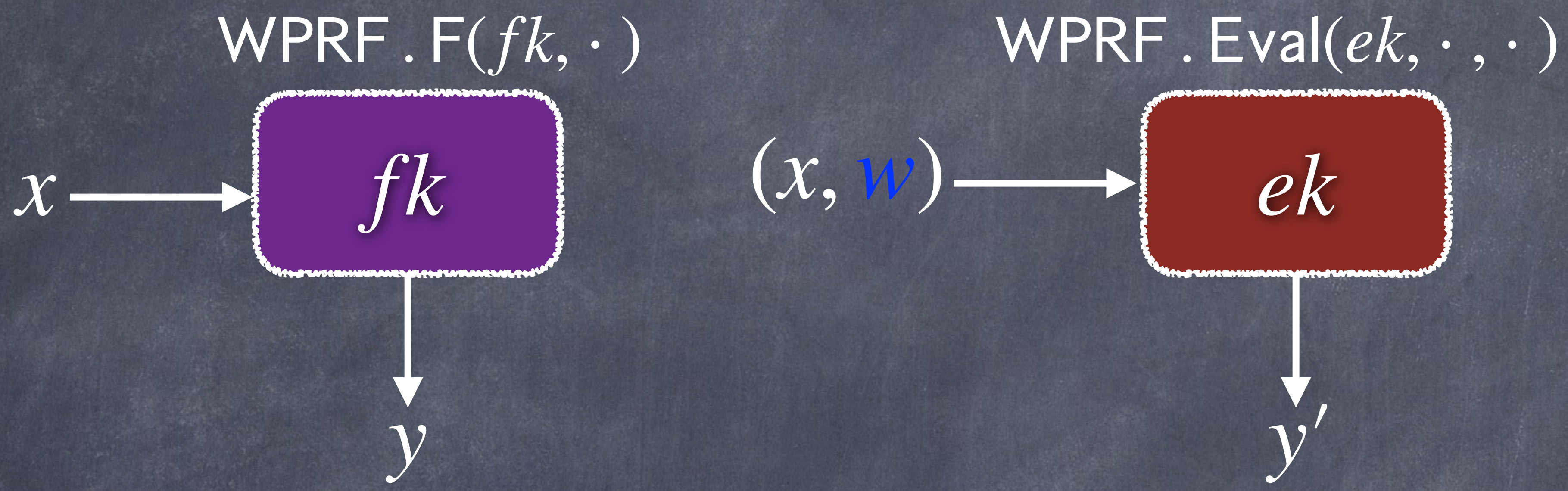
y

Equivalence of dv-UWM and Witness PRF





Equivalence of dv-UWM and Witness PRF



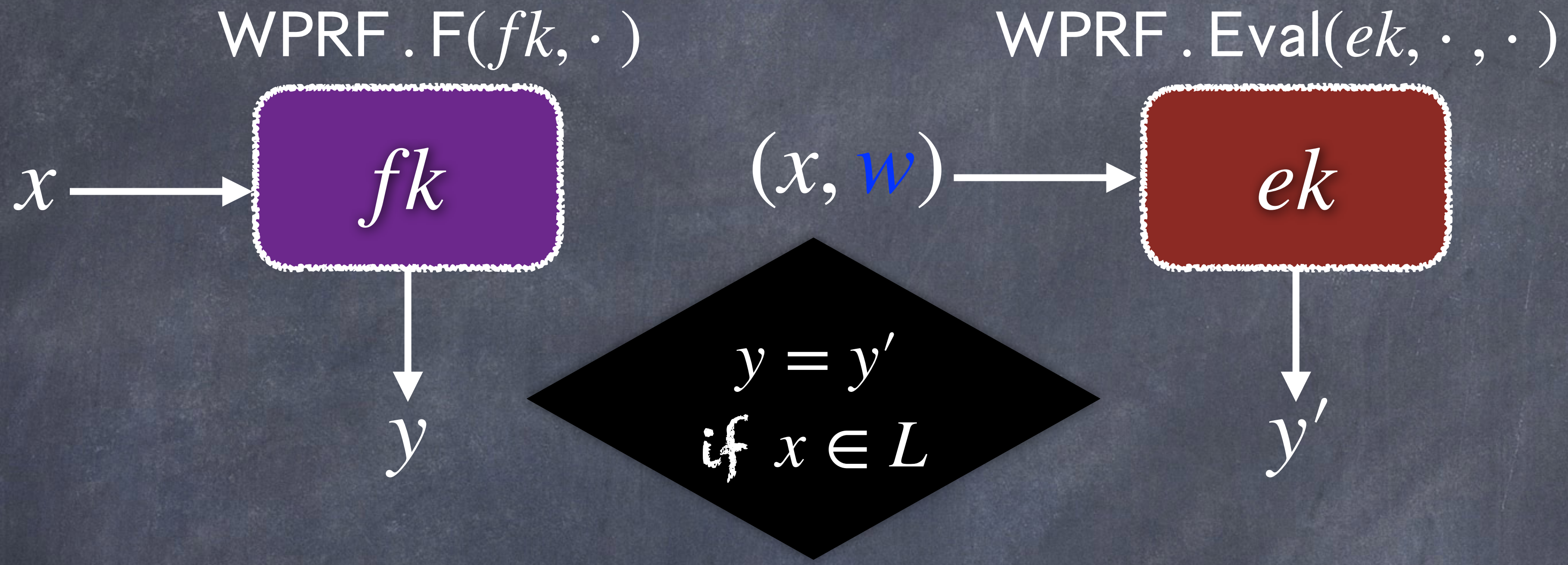
WPRF . F$(fk, \cdot\,)$

$x \rightarrow \boxed{fk} \rightarrow y$

WPRF . Eval$(ek, \cdot\,, \cdot\,)$

$(x, w) \rightarrow \boxed{ek} \rightarrow y'$

Equivalence of dv-UWM and Witness PRF



WPRF . F$(fk, \cdot)$

WPRF . Eval$(ek, \cdot, \cdot)$

x

fk

(x, w)

ek

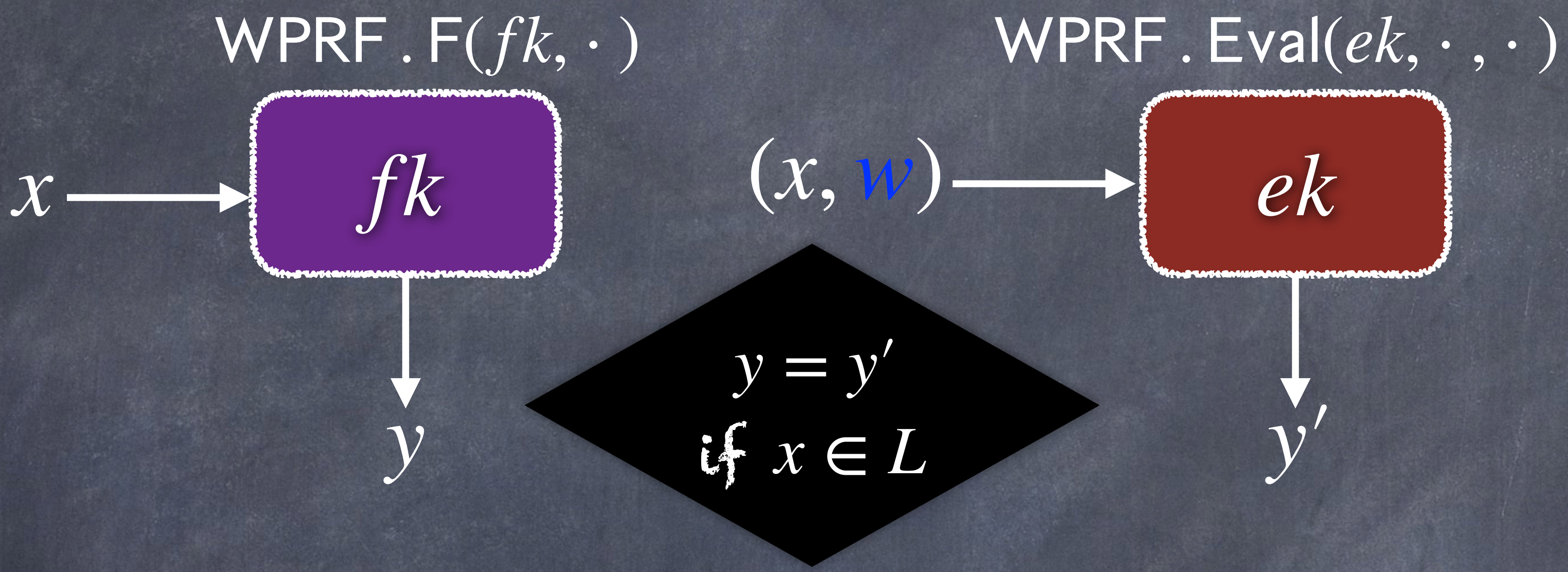
y

$y = y'$

if $x \in L$

y'

Equivalence of dv-UWM and Witness PRF



WPRF . F$(fk, \cdot)$

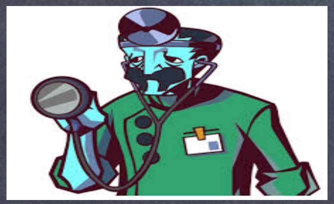
WPRF . Eval$(ek, \cdot, \cdot)$

$x \rightarrow fk \rightarrow y$

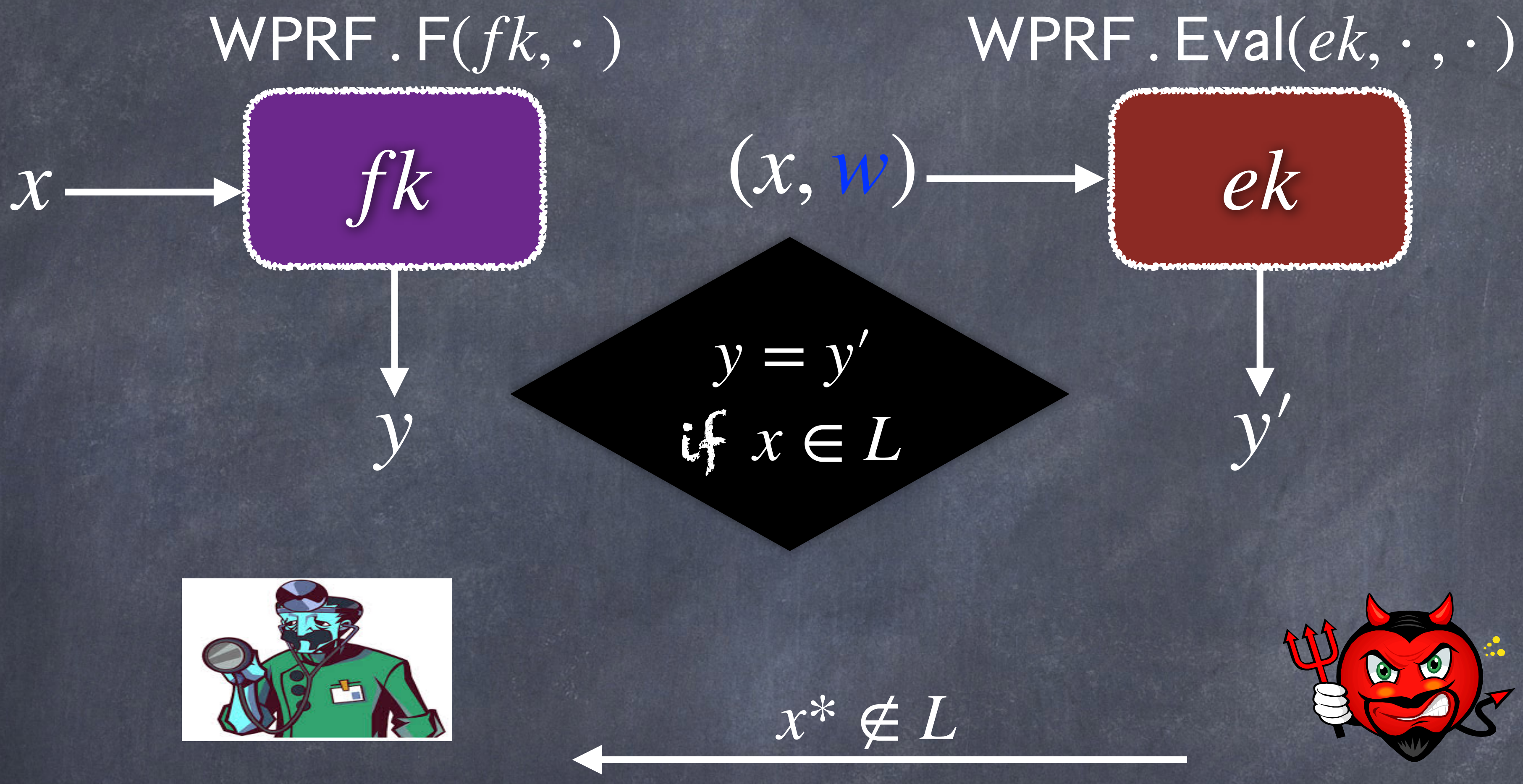
$(x, w) \rightarrow ek \rightarrow y'$

$y = y'$

if $x \in L$



Equivalence of dv-UWM and Witness PRF



$\mathsf{WPRF.F}(fk, \cdot)$

$\mathsf{WPRF.Eval}(ek, \cdot, \cdot)$

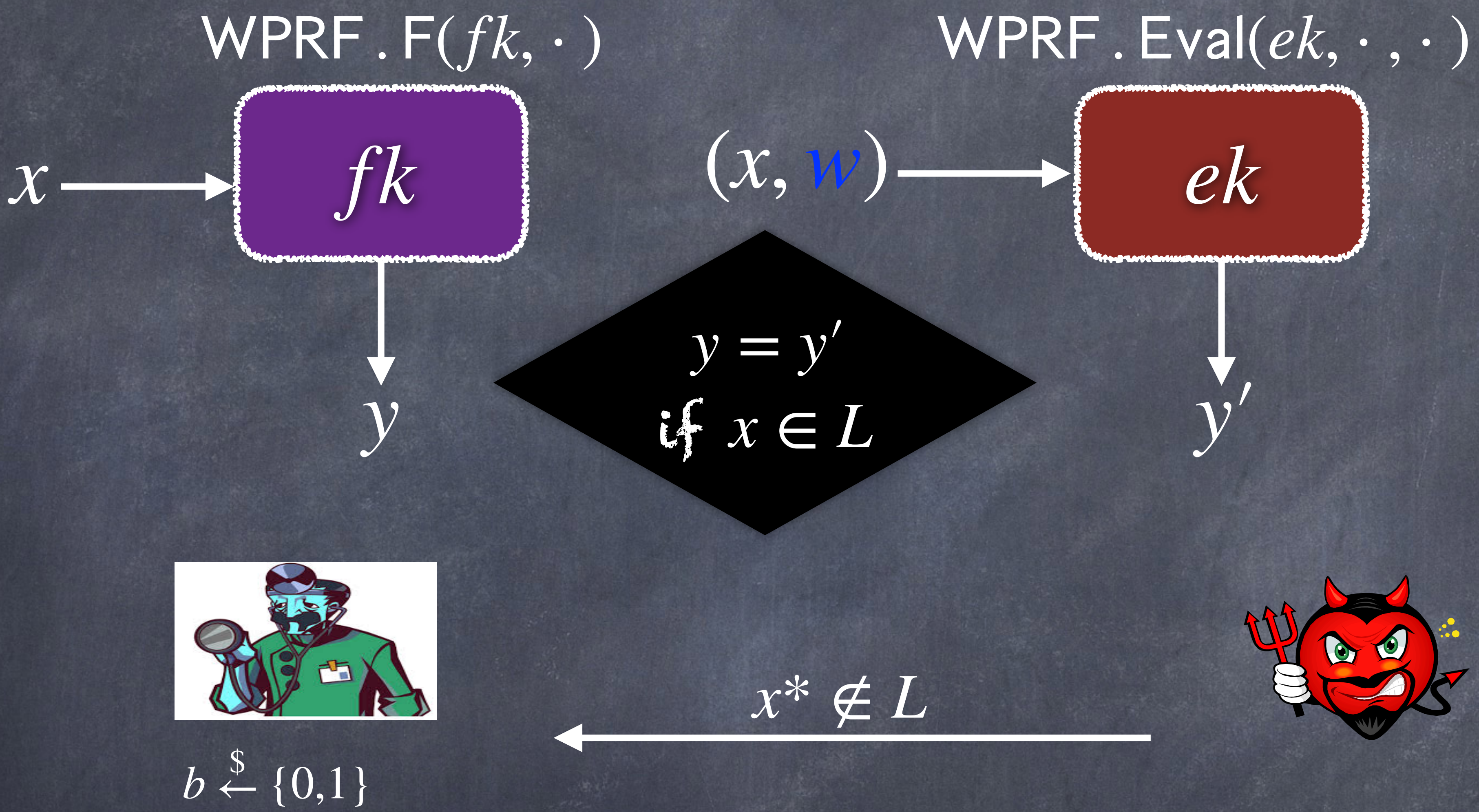
$x \rightarrow \boxed{fk} \rightarrow y$

$(x, w) \rightarrow \boxed{ek} \rightarrow y'$

$y = y'$
if $x \in L$

$x^* \notin L$

Equivalence of dv-UWM and Witness PRF

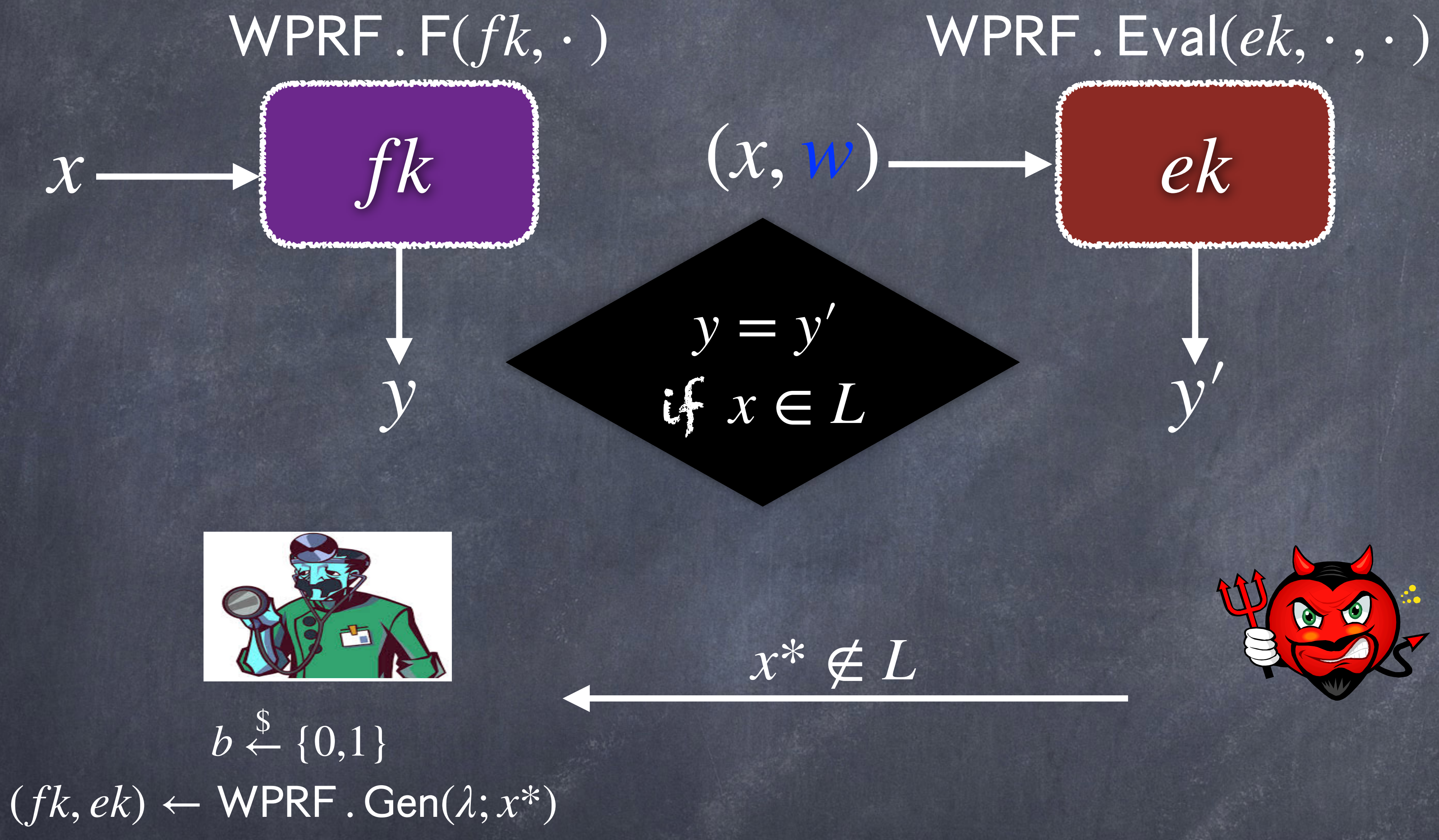


$\mathsf{WPRF.F}(fk, \cdot)$

$\mathsf{WPRF.Eval}(ek, \cdot, \cdot)$

$b \xleftarrow{\$} \{0,1\}$

Equivalence of dv-UWM and Witness PRF



$\mathsf{WPRF.F}(fk, \cdot)$

$\mathsf{WPRF.Eval}(ek, \cdot, \cdot)$

x

fk

(x, w)

ek

y

y'

$y = y'$

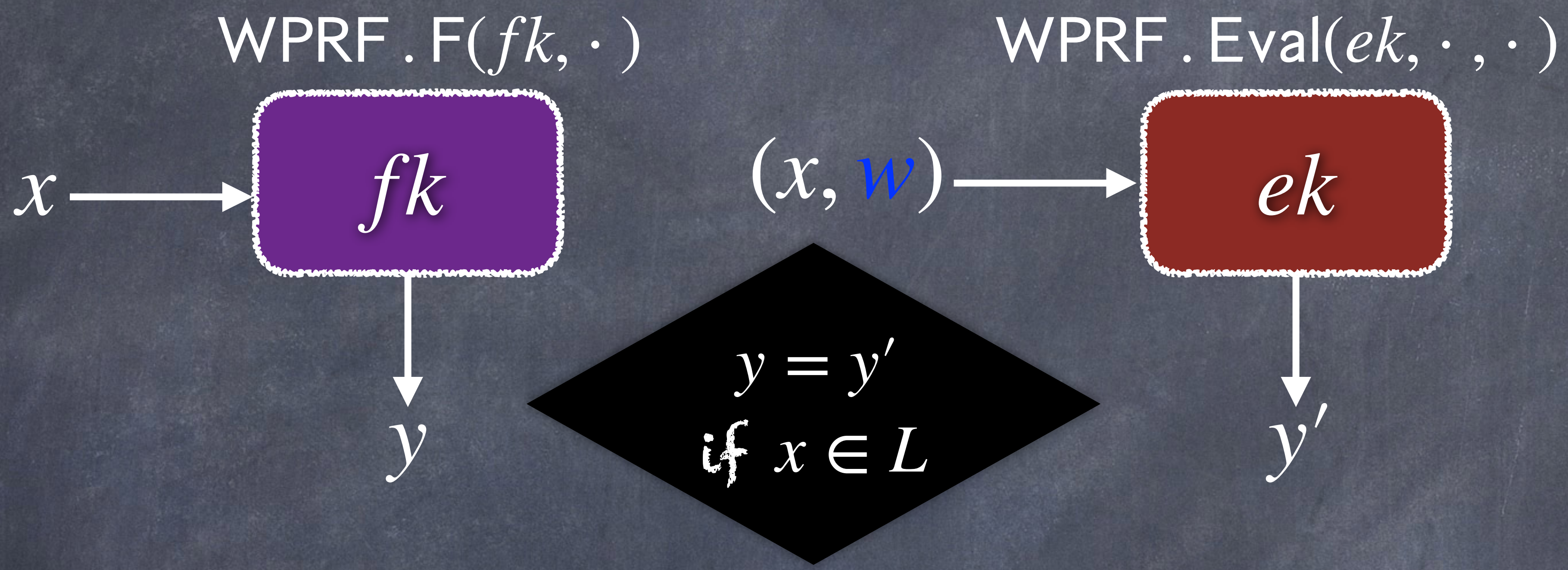
if $x \in L$

$x^* \notin L$

$b \xleftarrow{\$} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda; x^*)$

Equivalence of dv-UWM and Witness PRF



$\mathsf{WPRF.F}(fk, \cdot)$

$x \rightarrow \boxed{fk} \rightarrow y$

$\mathsf{WPRF.Eval}(ek, \cdot, \cdot)$

$(x, w) \rightarrow \boxed{ek} \rightarrow y'$

$y = y'$

if $x \in L$

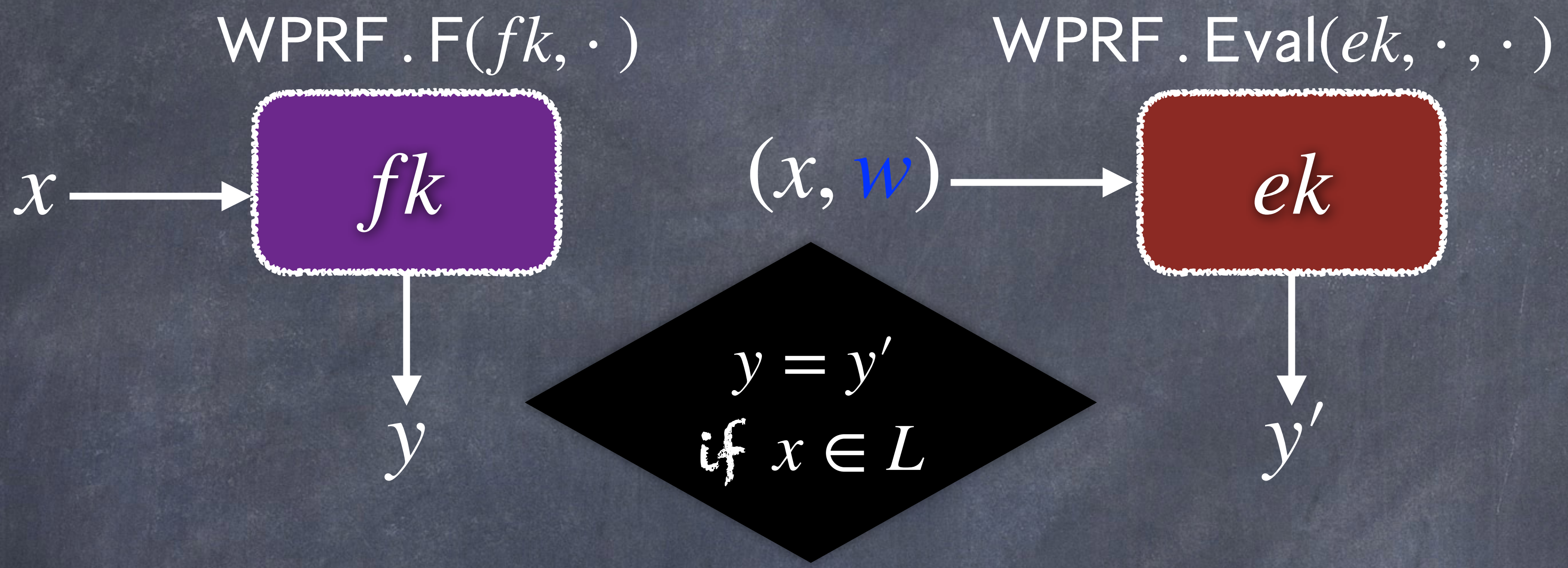
$b \overset{\$}{\leftarrow} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda; x^*)$

$\xleftarrow{x^* \notin L}$

$\xrightarrow{ek}$

Equivalence of dv-UWM and Witness PRF



WPRF . F$(fk, \cdot)$

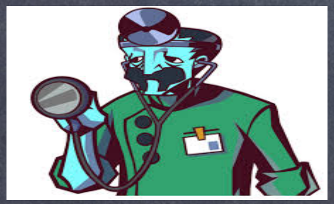
$x \rightarrow$ fk $\rightarrow y$

WPRF . Eval$(ek, \cdot, \cdot)$

$(x, w) \rightarrow$ ek $\rightarrow y'$

$y = y'$

if $x \in L$



$b \xleftarrow{\$} \{0,1\}$

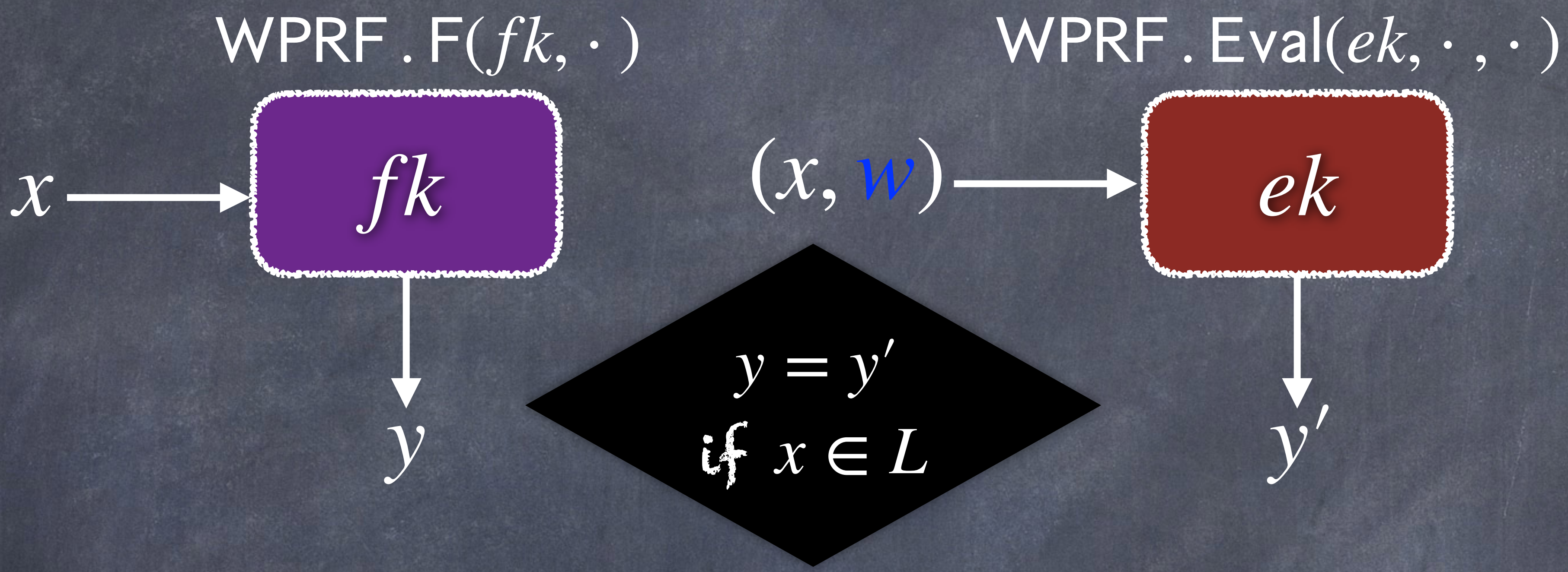
$(fk, ek) \leftarrow$ WPRF . Gen$(\lambda; x^*)$

$\xleftarrow{\quad x^* \notin L \quad}$

$\xrightarrow{\quad ek \quad}$

$\xleftarrow{\quad x_i \quad}$

Equivalence of dv-UWM and Witness PRF



$\mathsf{WPRF}.\mathsf{F}(fk,\cdot)$

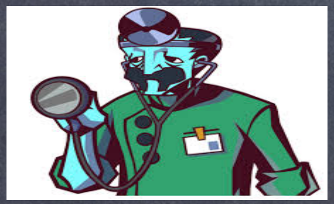
$\mathsf{WPRF}.\mathsf{Eval}(ek,\cdot,\cdot)$

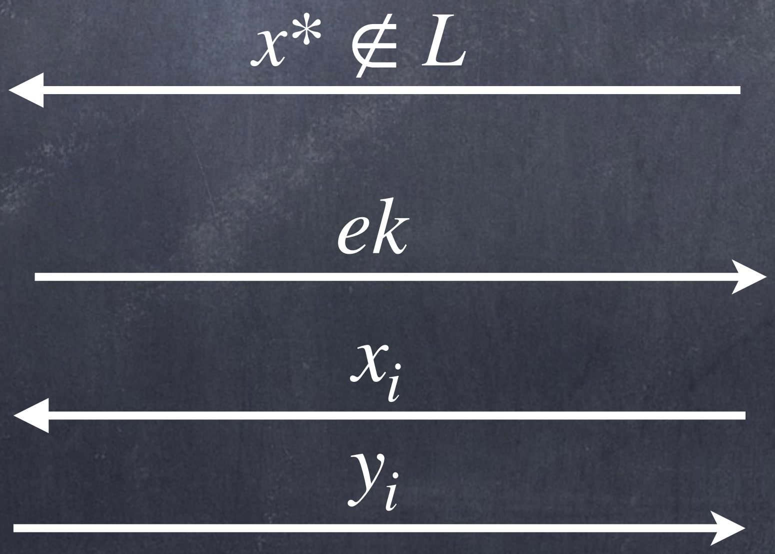
$x \rightarrow \boxed{fk} \rightarrow y$

$(x, w) \rightarrow \boxed{ek} \rightarrow y'$

$y = y'$

if $x \in L$





$b \overset{\$}{\leftarrow} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF}.\mathsf{Gen}(\lambda; x^*)$

$y_i \leftarrow \mathsf{WPRF}.\mathsf{F}(fk, x_i)$

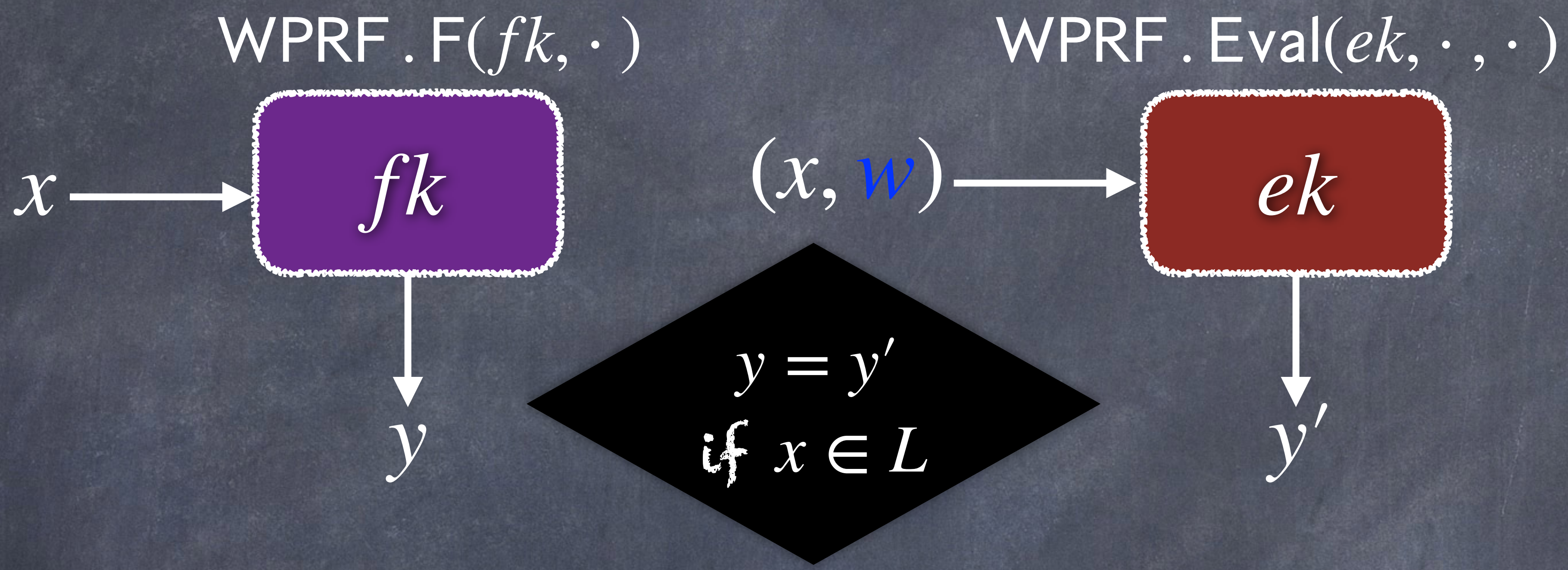
$\xleftarrow{x^* \notin L}$

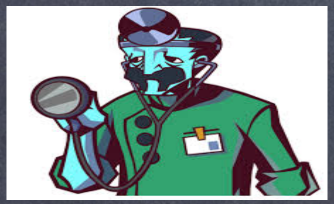
$\xrightarrow{ek}$

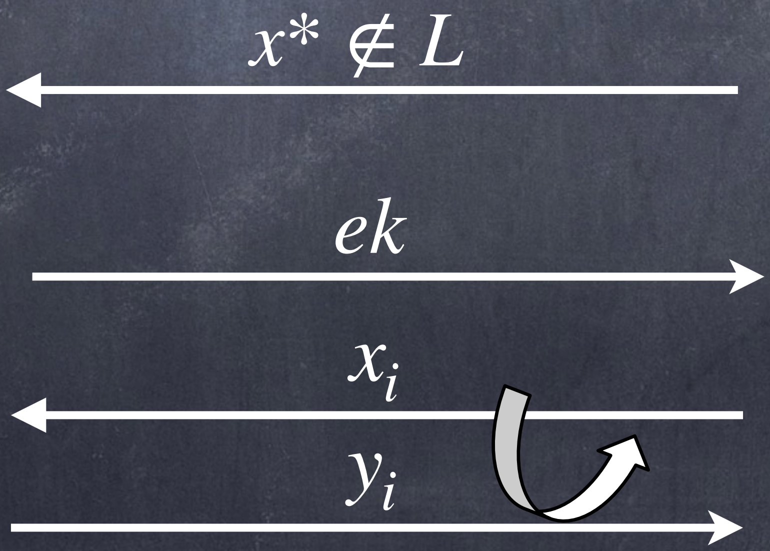
$\xleftarrow{x_i}$

$\xrightarrow{y_i}$

Equivalence of dv-UWM and Witness PRF





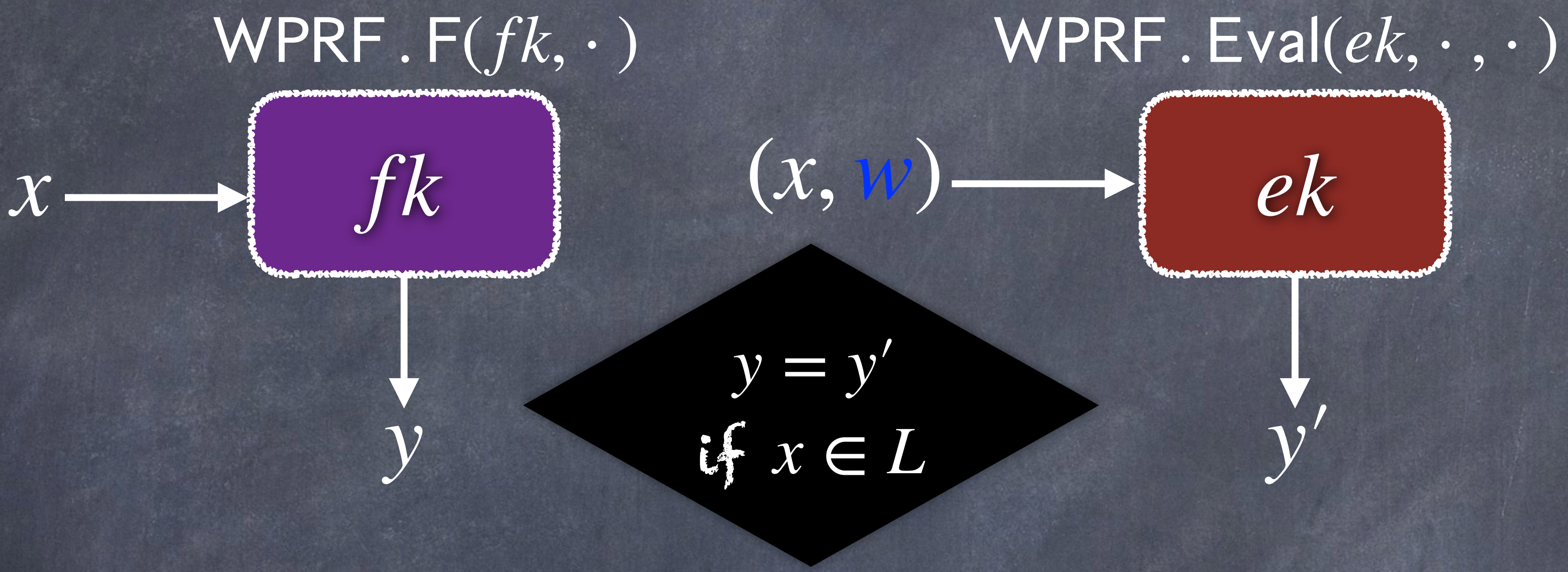


$b \overset{\$}{\leftarrow} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda; x^*)$

$y_i \leftarrow \mathsf{WPRF.F}(fk, x_i)$

Equivalence of dv-UWM and Witness PRF



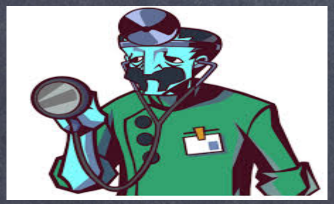
$\mathsf{WPRF.F}(fk, \cdot)$

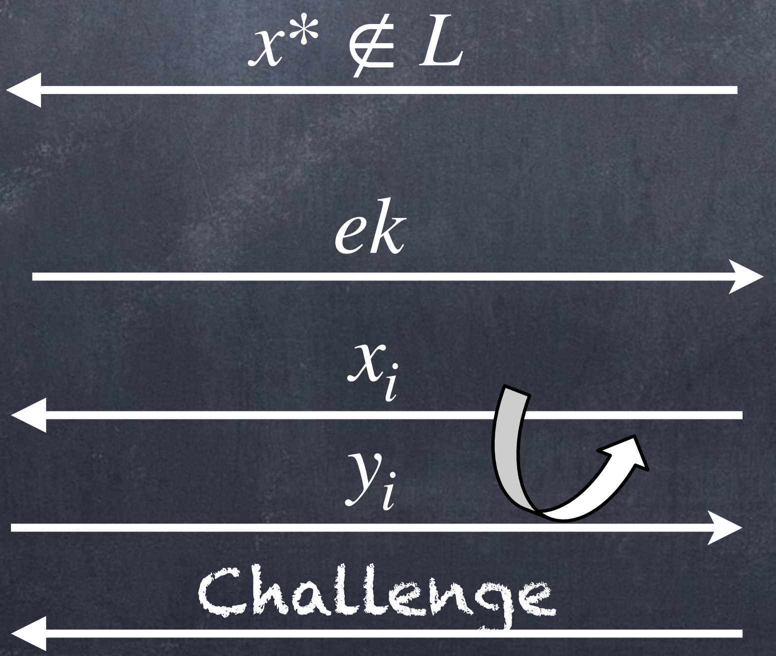
$x \longrightarrow \boxed{fk} \longrightarrow y$

$\mathsf{WPRF.Eval}(ek, \cdot, \cdot)$

$(x, w) \longrightarrow \boxed{ek} \longrightarrow y'$

$y = y'$ if $x \in L$





$x^* \notin L$

$b \overset{\$}{\leftarrow} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda; x^*)$

ek

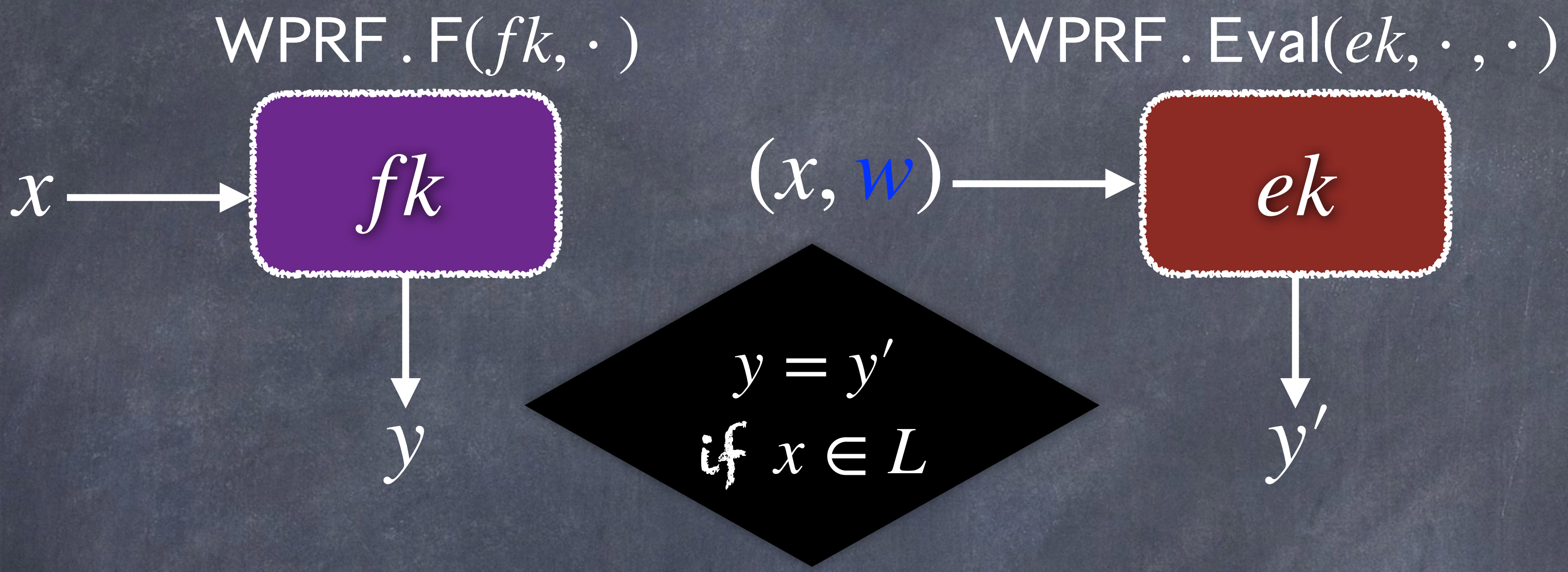
x_i

$y_i \leftarrow \mathsf{WPRF.F}(fk, x_i)$

y_i

Challenge

Equivalence of dv-UWM and Witness PRF



$\mathsf{WPRF.F}(fk, \cdot)$

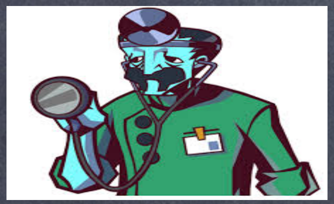
$\mathsf{WPRF.Eval}(ek, \cdot, \cdot)$

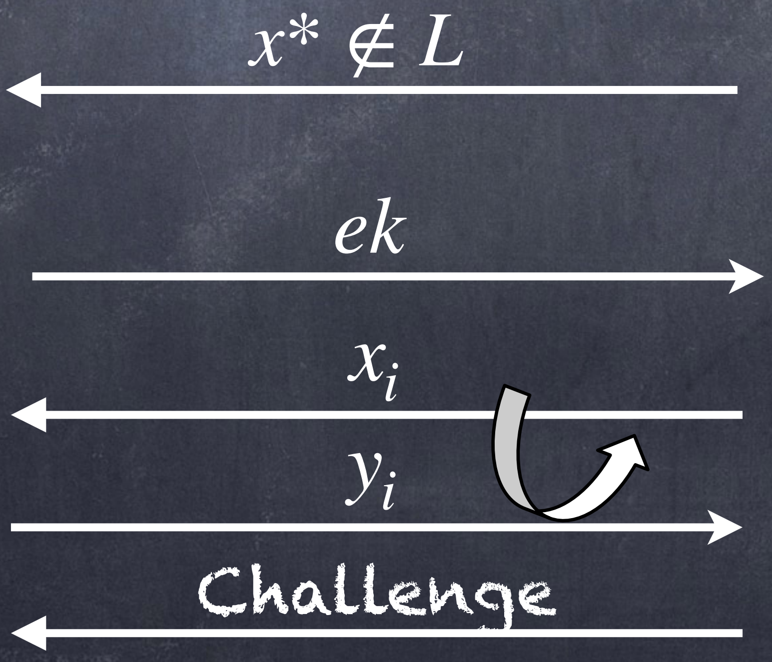
$x \rightarrow \boxed{fk} \rightarrow y$

$(x, w) \rightarrow \boxed{ek} \rightarrow y'$

$y = y'$

if $x \in L$





$x^* \notin L$

$b \overset{\$}{\leftarrow} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda; x^*)$

ek

x_i

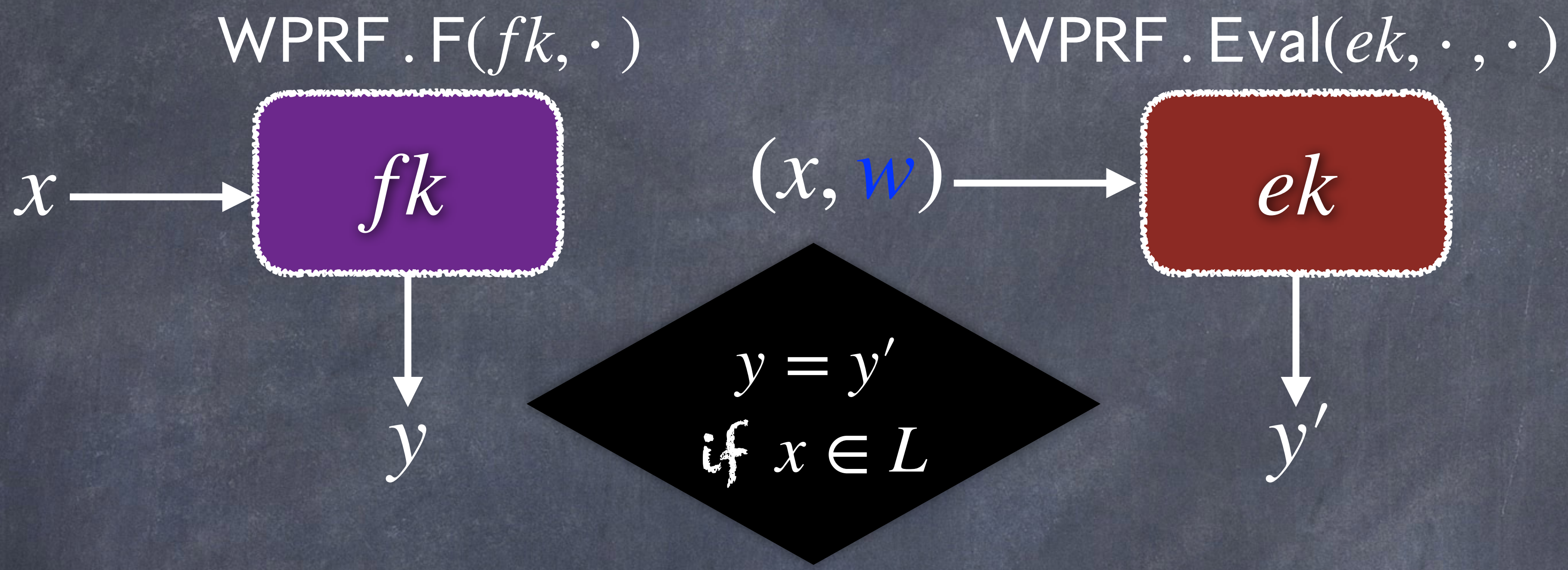
$y_i \leftarrow \mathsf{WPRF.F}(fk, x_i)$

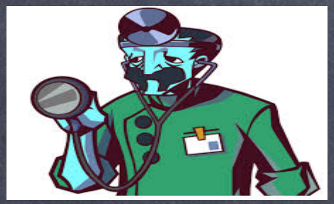
y_i

Challenge

$y_0 \leftarrow \mathsf{WPRF.F}(fk, x^*)$

Equivalence of dv-UWM and Witness PRF





$x^* \notin L$

$b \xleftarrow{\$} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda; x^*)$

ek

x_i

$y_i \leftarrow \mathsf{WPRF.F}(fk, x_i)$

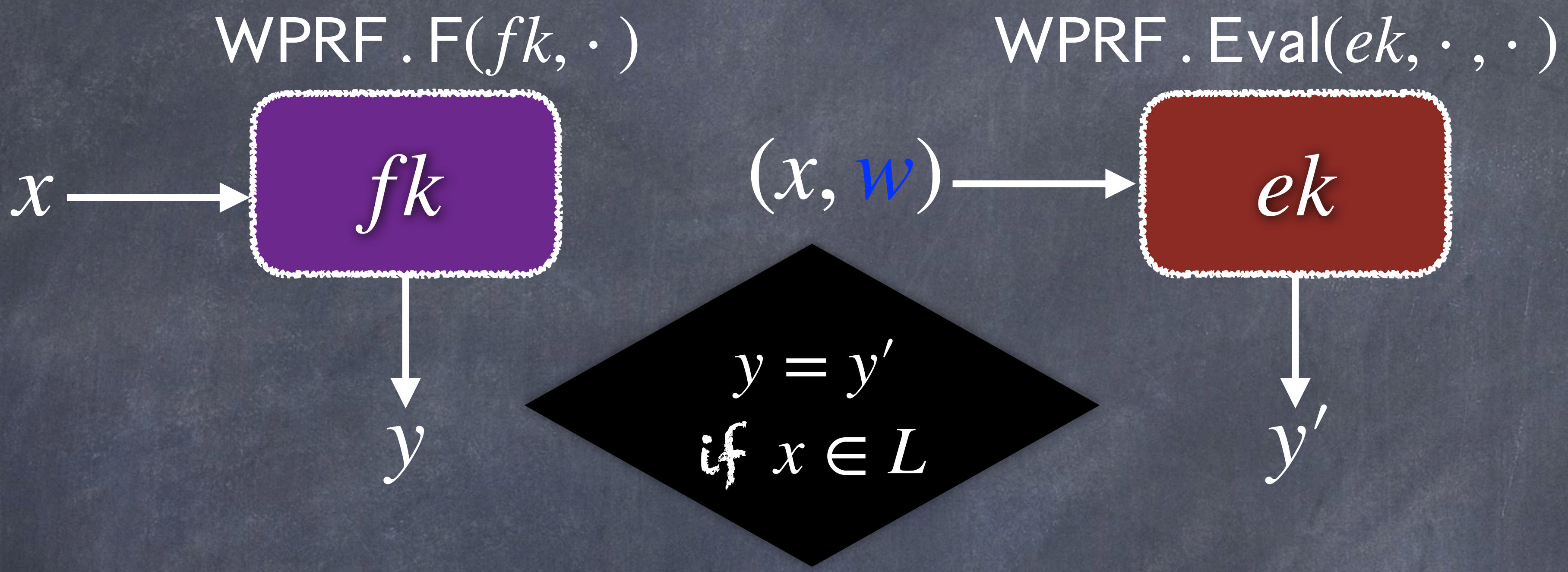
y_i

Challenge

$y_0 \leftarrow \mathsf{WPRF.F}(fk, x^*)$

$y_1 \xleftarrow{\$} \mathcal{Y}$

Equivalence of dv-UWM and Witness PRF



$\mathsf{WPRF}.\mathsf{F}(fk, \cdot)$

$\mathsf{WPRF}.\mathsf{Eval}(ek, \cdot, \cdot)$

$x \rightarrow \boxed{fk} \rightarrow y$

$(x, w) \rightarrow \boxed{ek} \rightarrow y'$

$y = y'$
if $x \in L$

$\xleftarrow{x^* \notin L}$

$b \xleftarrow{\$} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF}.\mathsf{Gen}(\lambda; x^*)$

$\xrightarrow{ek}$

$\xleftarrow{x_i}$

$y_i \leftarrow \mathsf{WPRF}.\mathsf{F}(fk, x_i)$

$\xrightarrow{y_i}$

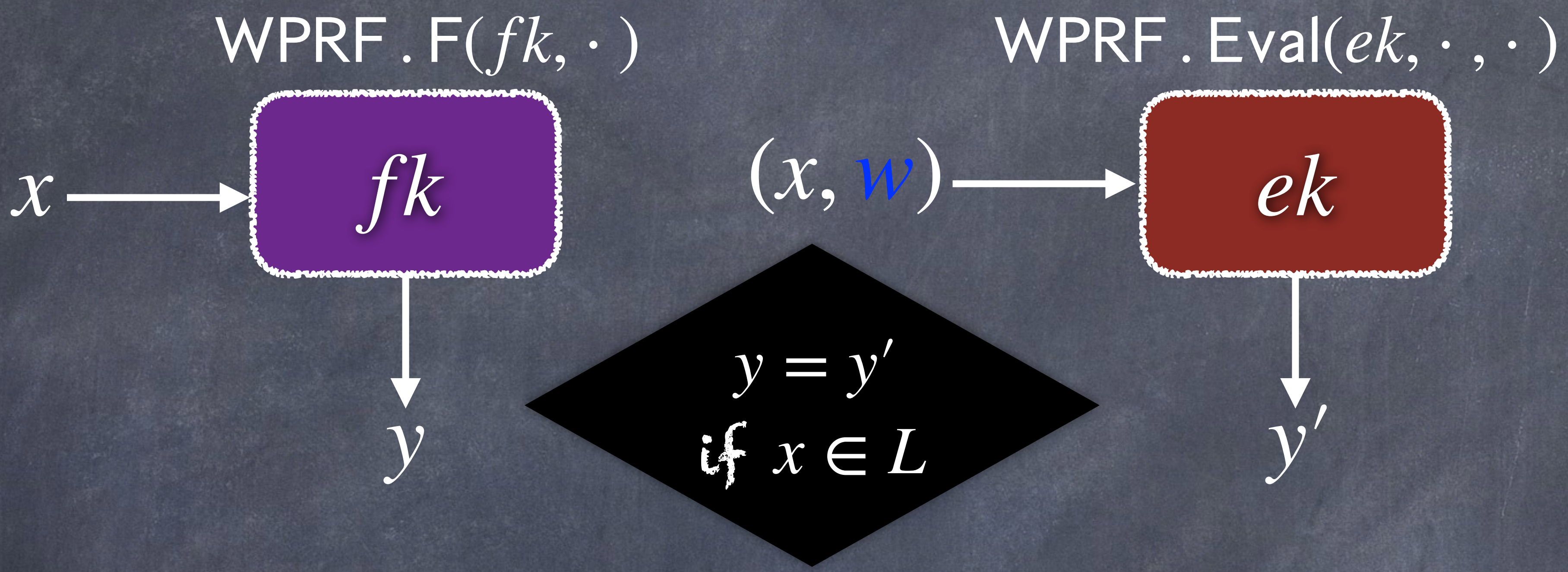
$\xleftarrow{\text{Challenge}}$

$y_0 \leftarrow \mathsf{WPRF}.\mathsf{F}(fk, x^*)$

$y_1 \xleftarrow{\$} \mathcal{Y}$

$\xrightarrow{y_b}$

Equivalence of dv-UWM and Witness PRF



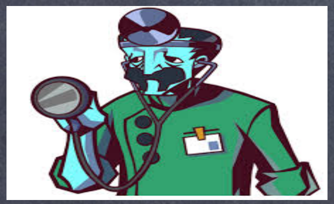
$\mathsf{WPRF}.\mathsf{F}(fk, \cdot)$

$x \longrightarrow \boxed{fk} \longrightarrow y$

$\mathsf{WPRF}.\mathsf{Eval}(ek, \cdot, \cdot)$

$(x, w) \longrightarrow \boxed{ek} \longrightarrow y'$

$y = y'$
if $x \in L$



$\xleftarrow{x^* \notin L}$

$b \xleftarrow{\$} \{0,1\}$
$(fk, ek) \leftarrow \mathsf{WPRF}.\mathsf{Gen}(\lambda; x^*)$

$\xrightarrow{ek}$

$\xleftarrow{x_i}$

$y_i \leftarrow \mathsf{WPRF}.\mathsf{F}(fk, x_i)$

$\xrightarrow{y_i}$

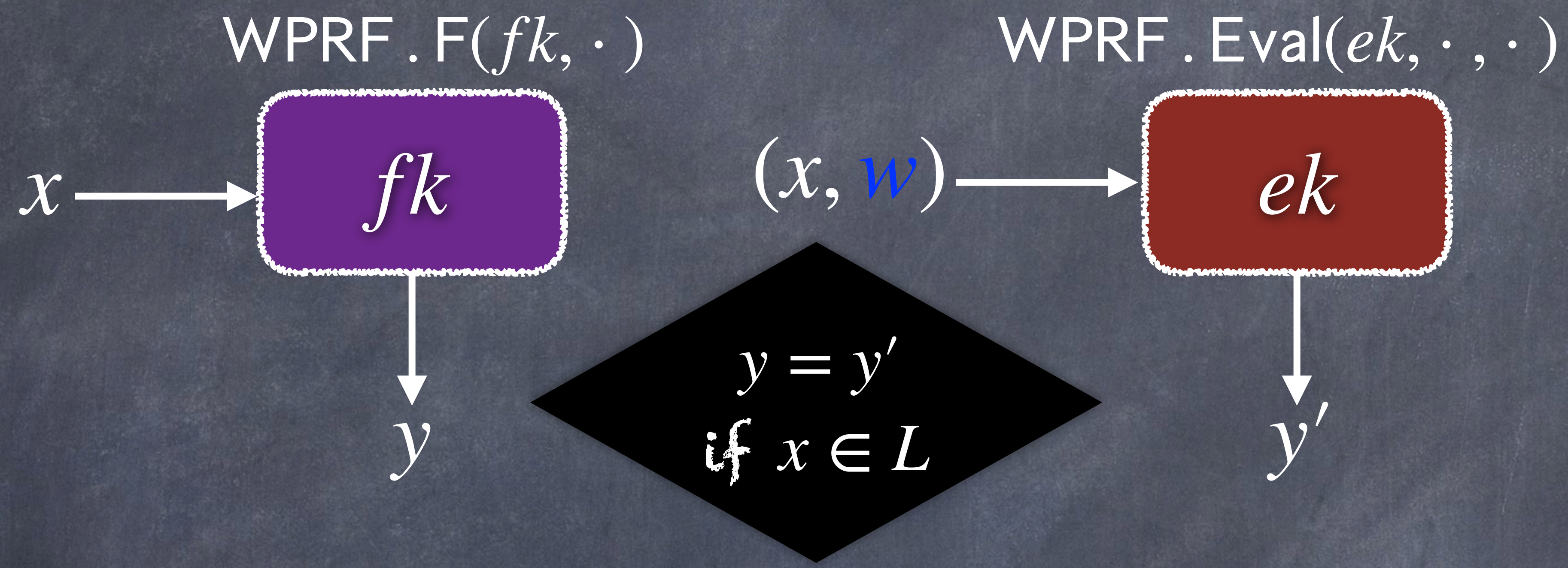
$\xleftarrow{\text{Challenge}}$

$y_0 \leftarrow \mathsf{WPRF}.\mathsf{F}(fk, x^*)$
$y_1 \xleftarrow{\$} \mathcal{Y}$

$\xrightarrow{y_b}$

Wins if guesses bit b
w.h.p

Equivalence of dv-UWM and Witness PRF



$\mathsf{WPRF.F}(fk, \cdot)$

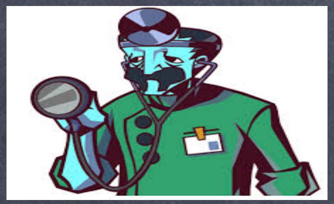
$x \rightarrow \boxed{fk} \rightarrow y$

$\mathsf{WPRF.Eval}(ek, \cdot, \cdot)$

$(x, w) \rightarrow \boxed{ek} \rightarrow y'$

$y = y'$

if $x \in L$



$x^* \notin L$ ←

$b \xleftarrow{\$} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda; x^*)$

→ ek

← x_i

$y_i \leftarrow \mathsf{WPRF.F}(fk, x_i)$

→ y_i

(Static-instance) Interactive security of WPRF

← Challenge

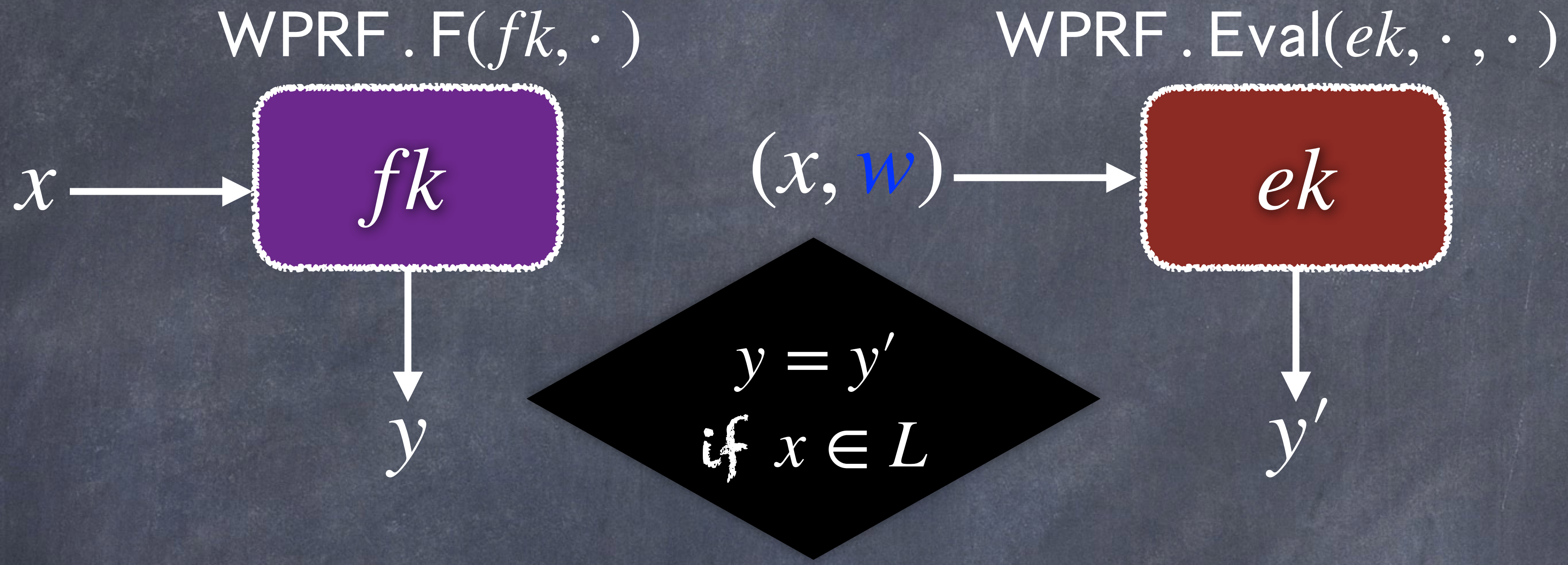
$y_0 \leftarrow \mathsf{WPRF.F}(fk, x^*)$

$y_1 \xleftarrow{\$} \mathcal{Y}$

→ y_b

Wins if guesses bit b w.h.p

Equivalence of dv-UWM and Witness PRF



$\mathsf{WPRF}.\mathsf{F}(fk, \cdot)$

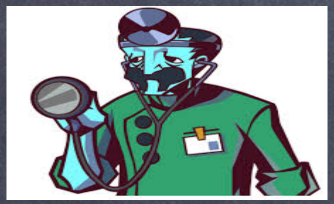
$x \rightarrow fk \rightarrow y$

$\mathsf{WPRF}.\mathsf{Eval}(ek, \cdot, \cdot)$

$(x, w) \rightarrow ek \rightarrow y'$

$y = y'$

if $x \in L$



$x^* \notin L$

$b \xleftarrow{\$} \{0,1\}$

$(fk, ek) \leftarrow \mathsf{WPRF}.\mathsf{Gen}(\lambda; x^*)$

ek

(Static-instance) Non-Interactive (nI) security of WPRF

$y_i \leftarrow \mathsf{WPRF}.\mathsf{F}(fk, x_i)$

Challenge

$y_0 \leftarrow \mathsf{WPRF}.\mathsf{F}(fk, x^*)$

$y_1 \xleftarrow{\$} \mathcal{Y}$

y_b

Wins if guesses bit b w.h.p

Witness PRF => DV-UWM

Witness PRF => DV-UWM

1. dv . setup(λ): Run $(fk, ek) \leftarrow$ WPRF . Gen(λ);

Witness PRF => DV-UWM

1. dv . setup(λ): Run $(fk, ek) \leftarrow$ WPRF . Gen(λ);
   - Set $crs = ek$ and $VK = fk$

Witness PRF => DV-UWM

1. $\mathsf{dv}.\mathsf{setup}(\lambda)$: Run $(fk, ek) \leftarrow \mathsf{WPRF}.\mathsf{Gen}(\lambda)$;
   - Set $crs = ek$ and $VK = fk$
2. $\mathsf{dv}.\mathsf{map}(crs, x, w)$: Run $y \leftarrow \mathsf{WPRF}.\mathsf{Eval}(ek, x, w)$; set $w* = y$

Witness PRF => DV-UWM

1. $\mathsf{dv.setup}(\lambda)$: Run $(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda)$;
   - Set $crs = ek$ and $VK = fk$
2. $\mathsf{dv.map}(crs, x, w)$: Run $y \leftarrow \mathsf{WPRF.Eval}(ek, x, w)$; set $w^* = y$
3. $\mathsf{dv.check}(VK, x, w^*)$: Run $y' \leftarrow \mathsf{WPRF.F}(fk, x)$ and check if $w^* \stackrel{?}{=} y$. If so, output 1, else o/p 0.

Witness PRF => DV-UWM

1. dv . setup(λ): Run $(fk, ek) \leftarrow$ WPRF . Gen(λ);
    - Set $crs = ek$ and $VK = fk$
2. dv . map(crs, x, w): Run $y \leftarrow$ WPRF . Eval(ek, x, w); set $w^* = y$
3. dv . check(VK, x, w^*): Run $y' \leftarrow$ WPRF . F(fk, x) and check if $w^* \overset{?}{=} y$. If so, output 1, else o/p 0.

Proof of (selective) Reusable Soundness: Follows from (static-instance) interactive security of WPRF.

Witness PRF => DV-UWM

1. $\mathsf{dv.setup}(\lambda)$: Run $(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda)$;

   - Set $crs = ek$ and $VK = fk$

2. $\mathsf{dv.map}(crs, x, w)$: Run $y \leftarrow \mathsf{WPRF.Eval}(ek, x, w)$; set $w^* = y$

3. $\mathsf{dv.check}(VK, x, w^*)$: Run $y' \leftarrow \mathsf{WPRF.F}(fk, x)$ and check if $w^* \overset{?}{=} y$. If so, output 1, else o/p 0.

Proof of (selective) Reusable Soundness: Follows from (static-instance) interactive security of WPRF.

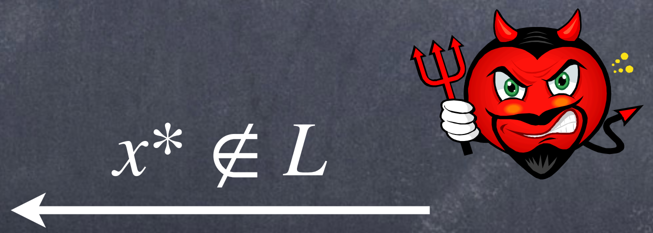
$x^* \notin L$ $\longleftarrow$

$x^* \notin L$ $\longleftarrow$

Witness PRF => DV-UWM

1. $\mathsf{dv.setup}(\lambda)$: Run $(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda)$;
   - Set $crs = ek$ and $VK = fk$
2. $\mathsf{dv.map}(crs, x, w)$: Run $y \leftarrow \mathsf{WPRF.Eval}(ek, x, w)$; set $w^* = y$
3. $\mathsf{dv.check}(VK, x, w^*)$: Run $y' \leftarrow \mathsf{WPRF.F}(fk, x)$ and check if $w^* \overset{?}{=} y$. If so, output 1, else o/p 0.

Proof of (selective) Reusable Soundness: Follows from (static-instance) interactive security of WPRF.



Witness PRF => DV-UWM

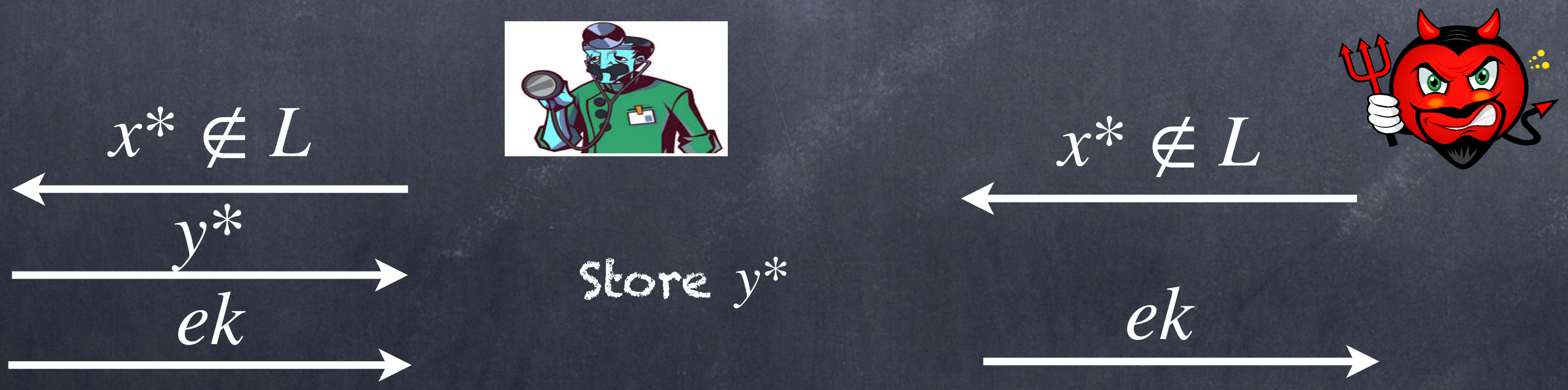
1. dv . setup(λ): Run $(fk, ek) \leftarrow$ WPRF . Gen(λ);
   - Set $crs = ek$ and $VK = fk$
2. dv . map(crs, x, w): Run $y \leftarrow$ WPRF . Eval(ek, x, w); set $w^* = y$
3. dv . check(VK, x, w^*): Run $y' \leftarrow$ WPRF . F(fk, x) and check if $w^* \overset{?}{=} y$. If so, output 1, else o/p 0.

Proof of (selective) Reusable Soundness: Follows from (static-instance) interactive security of WPRF.

Witness PRF => DV-UWM

1. $\mathsf{dv.setup}(\lambda)$: Run $(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda)$;
   - Set $crs = ek$ and $VK = fk$
2. $\mathsf{dv.map}(crs, x, w)$: Run $y \leftarrow \mathsf{WPRF.Eval}(ek, x, w)$; set $w^* = y$
3. $\mathsf{dv.check}(VK, x, w^*)$: Run $y' \leftarrow \mathsf{WPRF.F}(fk, x)$ and check if $w^* \overset{?}{=} y$. If so, output 1, else o/p 0.

Proof of (selective) Reusable Soundness: Follows from (static-instance) interactive security of WPRF.



Witness PRF => DV-UWM

1. $\mathsf{dv.setup}(\lambda)$: Run $(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda)$;
   - Set $crs = ek$ and $VK = fk$
2. $\mathsf{dv.map}(crs, x, w)$: Run $y \leftarrow \mathsf{WPRF.Eval}(ek, x, w)$; set $w^* = y$
3. $\mathsf{dv.check}(VK, x, w^*)$: Run $y' \leftarrow \mathsf{WPRF.F}(fk, x)$ and check if $w^* \overset{?}{=} y$. If so, output 1, else o/p 0.

Proof of (selective) Reusable Soundness: Follows from (static-instance) interactive security of WPRF.

$x^* \notin L$

y^*

ek

$\mathsf{WPRF.F}(\cdot, x_i)$

Store y^*

$x^* \notin L$

ek

(x_i, w_i)

Witness PRF => DV-UWM

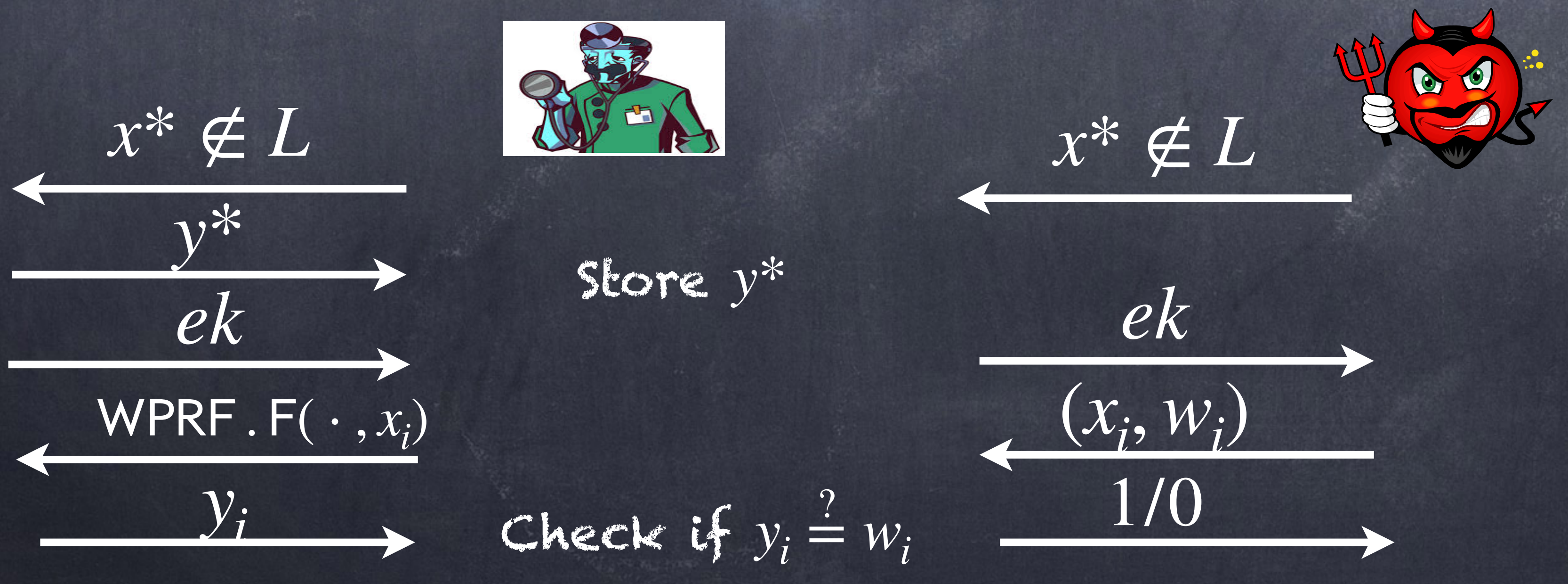
1. $\mathsf{dv.setup}(\lambda)$: Run $(fk, ek) \leftarrow \mathsf{WPRF.Gen}(\lambda)$;

   - Set $crs = ek$ and $VK = fk$

2. $\mathsf{dv.map}(crs, x, w)$: Run $y \leftarrow \mathsf{WPRF.Eval}(ek, x, w)$; set $w^* = y$

3. $\mathsf{dv.check}(VK, x, w^*)$: Run $y' \leftarrow \mathsf{WPRF.F}(fk, x)$ and check if $w^* \overset{?}{=} y$. If so, output 1, else o/p 0.

Proof of (selective) Reusable Soundness: Follows from (static-instance) interactive security of WPRF.



$x^* \notin L$

y^*

ek

$\mathsf{WPRF.F}(\cdot, x_i)$

y_i

Store y^*

Check if $y_i \overset{?}{=} w_i$

$x^* \notin L$

ek

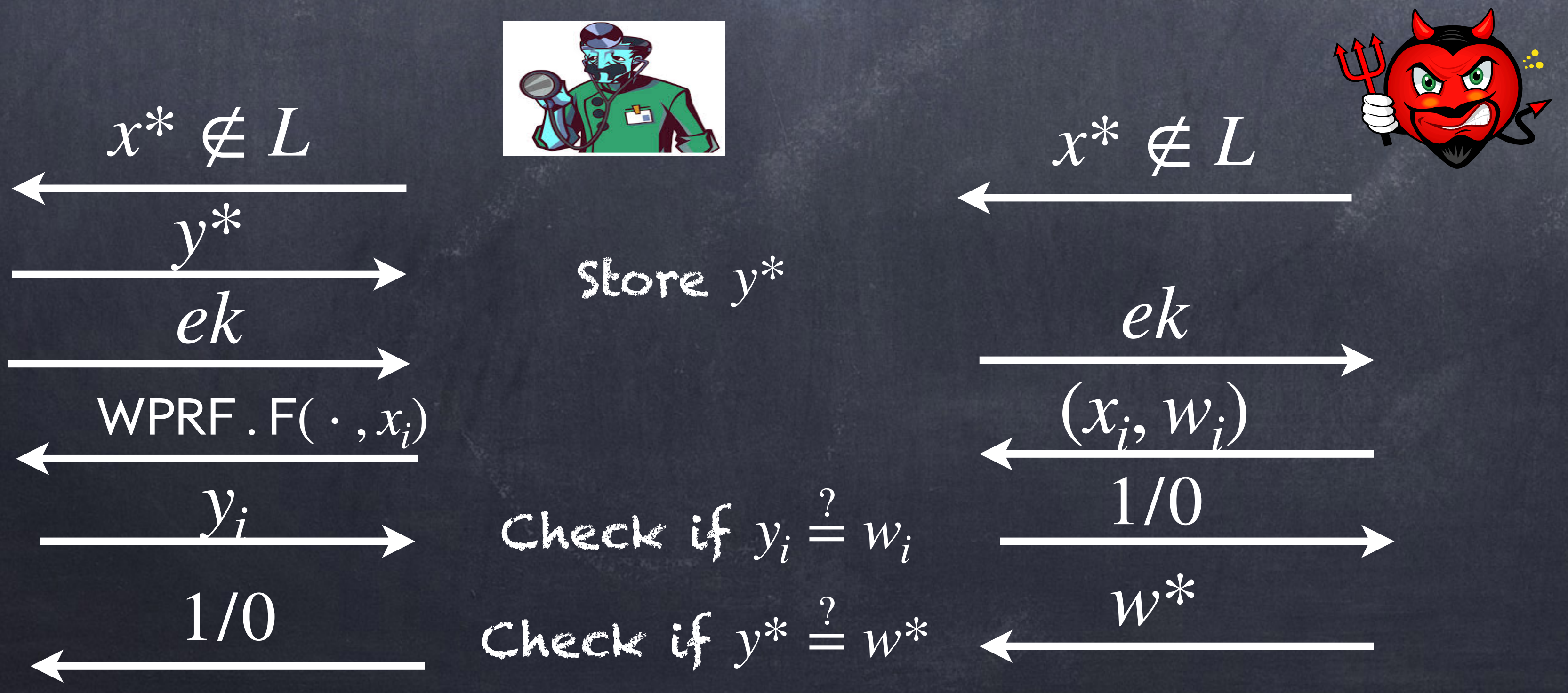
(x_i, w_i)

$1/0$

Witness PRF => DV-UWM

1. dv . setup(λ): Run $(fk, ek) \leftarrow$ WPRF . Gen(λ);
   - Set $crs = ek$ and $VK = fk$
2. dv . map(crs, x, w): Run $y \leftarrow$ WPRF . Eval(ek, x, w); set $w^* = y$
3. dv . check(VK, x, w^*): Run $y' \leftarrow$ WPRF . F(fk, x) and check if $w^* \overset{?}{=} y$. If so, output 1, else o/p 0.

Proof of (selective) Reusable Soundness: Follows from (static-instance) interactive security of WPRF.



$x^* \notin L$ ← | $x^* \notin L$ ←

y^* → | Store y^*

ek → | ek →

WPRF . F($\cdot$, x_i) ← | (x_i, w_i) ←

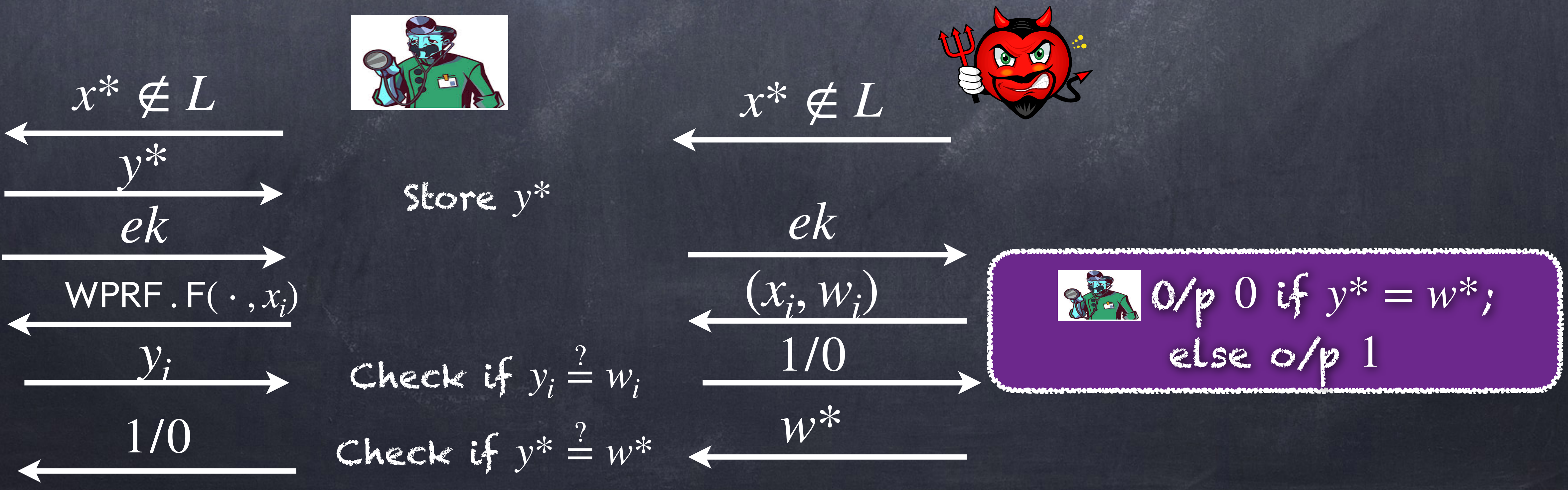
y_i → | Check if $y_i \overset{?}{=} w_i$ | $1/0$ →

$1/0$ ← | Check if $y^* \overset{?}{=} w^*$ | w^* ←

Witness PRF => DV-UWM

1. dv . setup(λ): Run $(fk, ek) \leftarrow$ WPRF . Gen(λ);
   - Set $crs = ek$ and $VK = fk$
2. dv . map(crs, x, w): Run $y \leftarrow$ WPRF . Eval(ek, x, w); set $w^* = y$
3. dv . check(VK, x, w^*): Run $y' \leftarrow$ WPRF . F(fk, x) and check if $w^* \overset{?}{=} y$. If so, output 1, else o/p 0.

Proof of (selective) Reusable Soundness: Follows from (static-instance) interactive security of WPRF.



$x^* \notin L$ ←

y^* →

ek →

WPRF . F($\cdot$, x_i) ←

y_i →

$1/0$ ←

Store y^*

Check if $y_i \overset{?}{=} w_i$

Check if $y^* \overset{?}{=} w^*$

← $x^* \notin L$

ek →

← (x_i, w_i)

$1/0$ →

← w^*

o/p 0 if $y^* = w^*$;
else o/p 1

(Non-reusable)DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(Non-reusable) DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

  $(crs, VK) \leftarrow$ dv . setup(λ)

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(Non-reusable) DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

$$(crs, VK) \leftarrow \boxed{\text{dv . setup}(\lambda)} \;;\quad z \in \{0,1\}^{\lambda} \longrightarrow \boxed{G} \longrightarrow y \in \{0,1\}^{2\lambda}$$

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(Non-reusable)DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

$$(crs, VK) \leftarrow \boxed{\text{dv . setup}(\lambda)}\ ; \quad z \in \{0,1\}^{\lambda} \rightarrow \boxed{G} \rightarrow y \in \{0,1\}^{2\lambda}$$

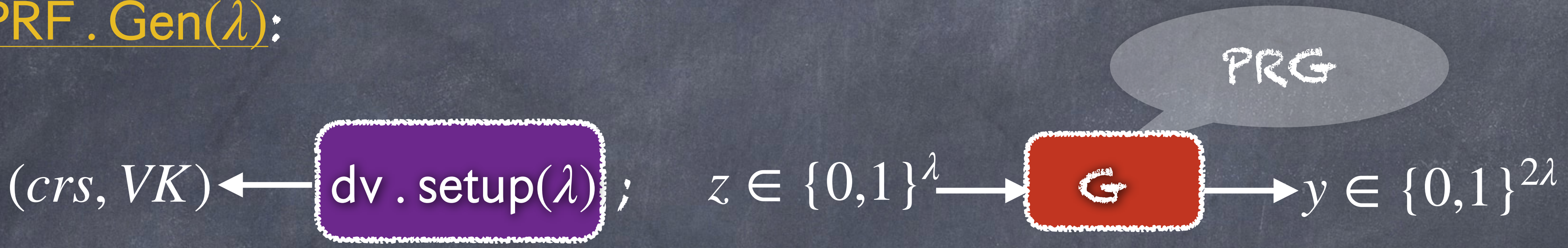
PRG

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(Non-reusable)DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):



$(crs, VK) \leftarrow \boxed{\text{dv . setup}(\lambda)}$; $z \in \{0,1\}^{\lambda} \rightarrow \boxed{G} \rightarrow y \in \{0,1\}^{2\lambda}$

PRG

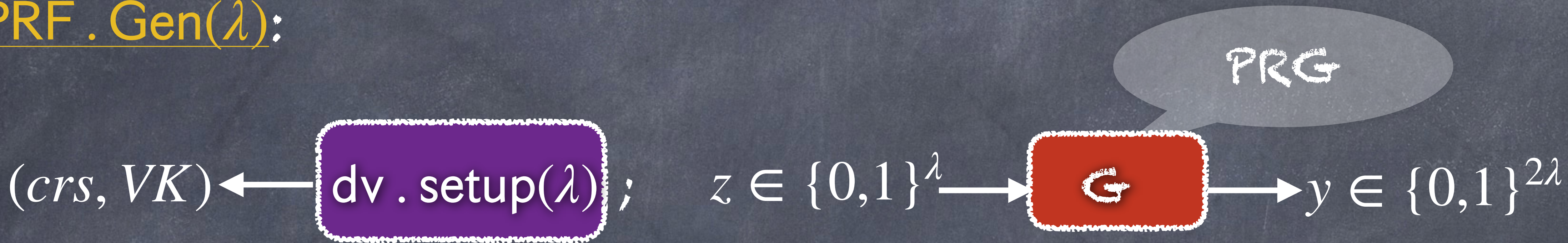
Set $ek = (crs, y)$ and $fk = (crs, z)$

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(Non-reusable)DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):



PRG

$(crs, VK) \leftarrow$ dv . setup(λ) ; $z \in \{0,1\}^{\lambda} \rightarrow$ G $\rightarrow y \in \{0,1\}^{2\lambda}$

Set $ek = (crs, y)$ and $fk = (crs, z)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR y is pseudorandom "

- WPRF . Eval(ek, x, w):

(Non-reusable)DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

PRG

$$(crs, VK) \leftarrow \boxed{\text{dv . setup}(\lambda)} \; ; \quad z \in \{0,1\}^{\lambda} \longrightarrow \boxed{G} \longrightarrow y \in \{0,1\}^{2\lambda}$$

Set $ek = (crs, y)$ and $fk = (crs, z)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR y is pseudorandom "

$$w^* \longleftarrow \boxed{\text{dv . map}(crs, \hat{x}, (\ \perp\ , z))}$$

- WPRF . Eval(ek, x, w):

(Non-reusable)DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

PRG

$$(crs, VK) \leftarrow \boxed{\text{dv . setup}(\lambda)} \;; \qquad z \in \{0,1\}^{\lambda} \rightarrow \boxed{G} \rightarrow y \in \{0,1\}^{2\lambda}$$

Set $ek = (crs, y)$ and $fk = (crs, z)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR y is pseudorandom "

$$w^* \leftarrow \boxed{\text{dv . map}(crs, \hat{x}, (\perp, z))} \;; \qquad b = GL(w^*)$$

- WPRF . Eval(ek, x, w):

(Non-reusable)DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

PRG

$$(crs, VK) \leftarrow \boxed{\text{dv . setup}(\lambda)}\ ; \quad z \in \{0,1\}^{\lambda} \rightarrow \boxed{G} \rightarrow y \in \{0,1\}^{2\lambda}$$

Set $ek = (crs, y)$ and $fk = (crs, z)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR y is pseudorandom "

$$w^* \leftarrow \boxed{\text{dv . map}(crs, \hat{x}, (\perp, z))}\ ; \quad b = GL(w^*)\ ; \quad \text{Set } y = b$$

- WPRF . Eval(ek, x, w):

(Non-reusable) DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

PRG

$$(crs, VK) \leftarrow \boxed{\text{dv . setup}(\lambda)} \; ; \qquad z \in \{0,1\}^{\lambda} \rightarrow \boxed{G} \rightarrow y \in \{0,1\}^{2\lambda}$$

Set $ek = (crs, y)$ and $fk = (crs, z)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR y is pseudorandom "

$$w^* \leftarrow \boxed{\text{dv . map}(crs, \hat{x}, (\perp, z))} \; ; \qquad b = GL(w^*) \; ; \quad \text{Set } y = b$$

- WPRF . Eval(ek, x, w):

$$\tilde{w}^* \leftarrow \boxed{\text{dv . map}(crs, \hat{x}, (w, \perp))}$$

(Non-reusable)DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

$$(crs, VK) \longleftarrow \boxed{\text{dv . setup}(\lambda)} \; ; \quad z \in \{0,1\}^{\lambda} \longrightarrow \boxed{G} \longrightarrow y \in \{0,1\}^{2\lambda}$$

PRG

Set $ek = (crs, y)$ and $fk = (crs, z)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR y is pseudorandom "

$$w^* \longleftarrow \boxed{\text{dv . map}(crs, \hat{x}, (\perp, z))} \; ; \quad b = GL(w^*) \; ; \quad \text{Set } y = b$$

- WPRF . Eval(ek, x, w):

$$\tilde{w}^* \longleftarrow \boxed{\text{dv . map}(crs, \hat{x}, (w, \perp))} \; ; \quad b' = GL(\tilde{w}^*)$$

(Non-reusable)DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

PRG

$$(crs, VK) \longleftarrow \boxed{\text{dv . setup}(\lambda)} \; ; \quad z \in \{0,1\}^{\lambda} \longrightarrow \boxed{G} \longrightarrow y \in \{0,1\}^{2\lambda}$$

Set $ek = (crs, y)$ and $fk = (crs, z)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR y is pseudorandom "

$$w^* \longleftarrow \boxed{\text{dv . map}(crs, \hat{x}, (\perp, z))} \; ; \quad b = GL(w^*) \; ; \quad \text{Set } y = b$$

- WPRF . Eval(ek, x, w):

$$\tilde{w}^* \longleftarrow \boxed{\text{dv . map}(crs, \hat{x}, (w, \perp))} \; ; \quad b' = GL(\tilde{w}^*) \; ; \quad \text{Set } y = b'$$

(Non-reusable) DV-UWM => (NI)-WPRF

- WPRF . Gen(λ):

PRG

$(crs, VK) \leftarrow \boxed{\text{dv . setup}(\lambda)}$; $z \in \{0,1\}^{\lambda} \rightarrow \boxed{G} \rightarrow y \in \{0,1\}^{2\lambda}$

Set $ek = (crs, y)$ and $fk = (crs, z)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR y is pseudorandom "

$w^* \leftarrow \boxed{\text{dv . map}(crs, \hat{x}, (\perp, z))}$; $b = GL(w^*)$; Set $y = b$

$w^* = \tilde{w}^*$

- WPRF . Eval(ek, x, w):

$\tilde{w}^* \leftarrow \boxed{\text{dv . map}(crs, \hat{x}, (w, \perp))}$; $b' = GL(\tilde{w}^*)$; Set $y = b'$

(Non-reusable) DV-UWM => (NI)-WPRF

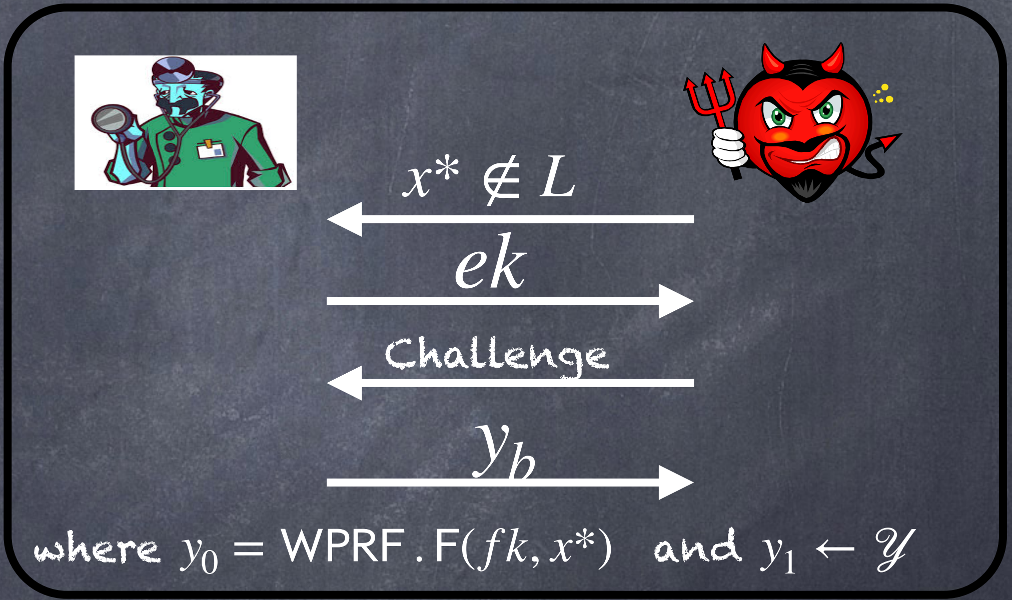
Intuition for Proof of (static-instance) Non-Interactive Security:

Recall $w^* \leftarrow \mathsf{dv.map}(crs, \hat{x}, (\perp, z))$ and $b = GL(w^*)$

(Non-reusable)DV-UWM => (NI)-WPRF

Intuition for Proof of (static-instance) Non-Interactive Security:

Recall $w^* \leftarrow \mathsf{dv.map}(crs, \hat{x}, (\perp, z))$ and $b = GL(w^*)$



$x^* \notin L$

ek

Challenge

y_b

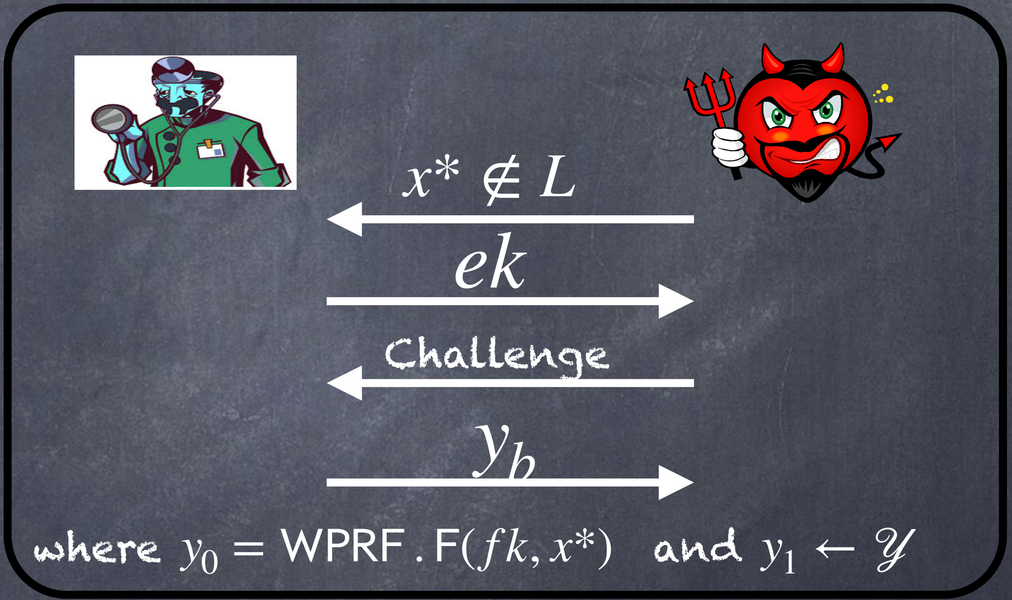
where $y_0 = \mathsf{WPRF.F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

(Non-reusable)DV-UWM => (NI)-WPRF

Intuition for Proof of (static-instance) Non-Interactive Security:

Recall $w^* \leftarrow \mathsf{dv.map}(crs, \hat{x}, (\perp, z))$ and $b = GL(w^*)$

- Convert a distinguisher for (NI)-WPRF to a predictor that outputs w^* w.h.p. s.t.
  - $\mathsf{dv.check}(crs, VK, \hat{x}, w^*) = 1$



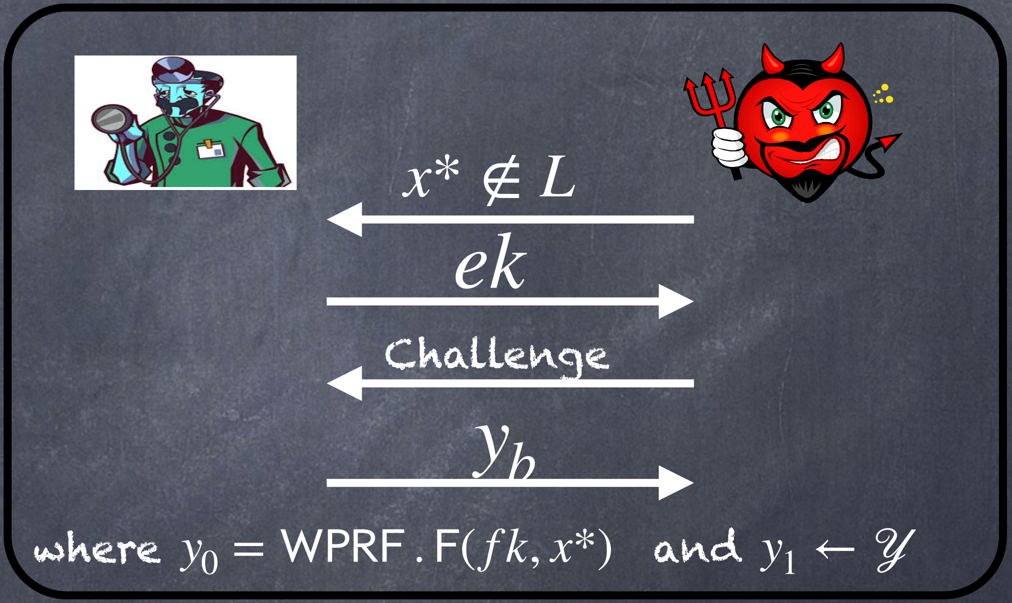
where $y_0 = \mathsf{WPRF.F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

(Non-reusable)DV-UWM => (NI)-WPRF

Intuition for Proof of (static-instance) Non-Interactive Security:

Recall $w^* \leftarrow \mathsf{dv.map}(crs, \hat{x}, (\perp, z))$ and $b = GL(w^*)$

- Convert a distinguisher for (NI)-WPRF to a predictor that outputs w^* w.h.p. s.t.
  - $\mathsf{dv.check}(crs, VK, \hat{x}, w^*) = 1$
- Switch $y \leftarrow \{0,1\}^{2\lambda}$ instead of $y = G(z)$.



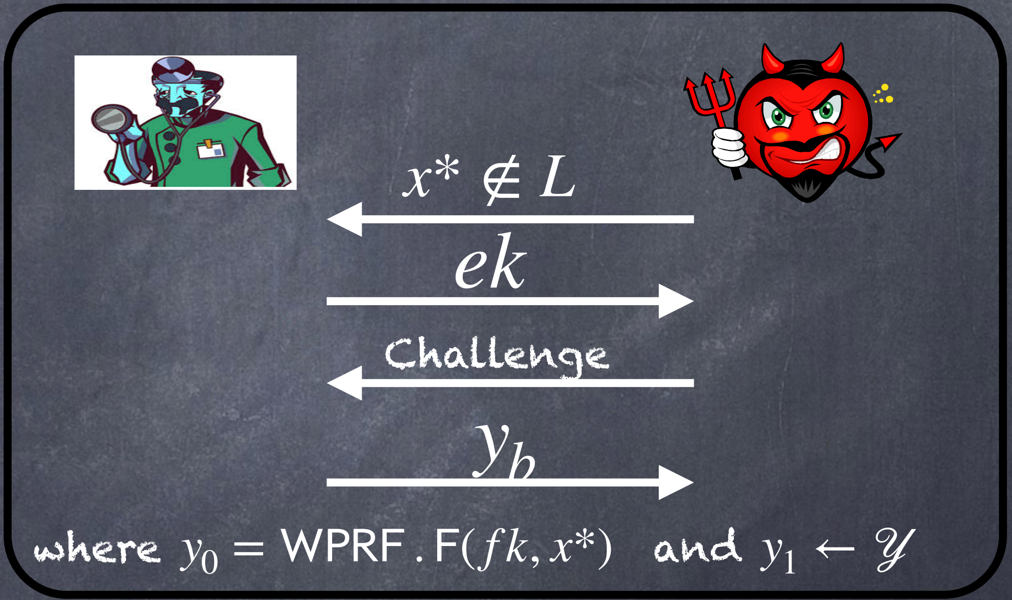
where $y_0 = \mathsf{WPRF.F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

(Non-reusable)DV-UWM => (NI)-WPRF

Intuition for Proof of (static-instance) Non-Interactive Security:

Recall $w^* \leftarrow \mathsf{dv.map}(crs, \hat{x}, (\perp, z))$ and $b = GL(w^*)$

- Convert a distinguisher for (NI)-WPRF to a predictor that outputs w^* w.h.p. s.t.
  - $\mathsf{dv.check}(crs, VK, \hat{x}, w^*) = 1$
- Switch $y \leftarrow \{0,1\}^{2\lambda}$ instead of $y = G(z)$.
- At this point the statement $\hat{x}$ is false; $x^* \notin L$ and y is random



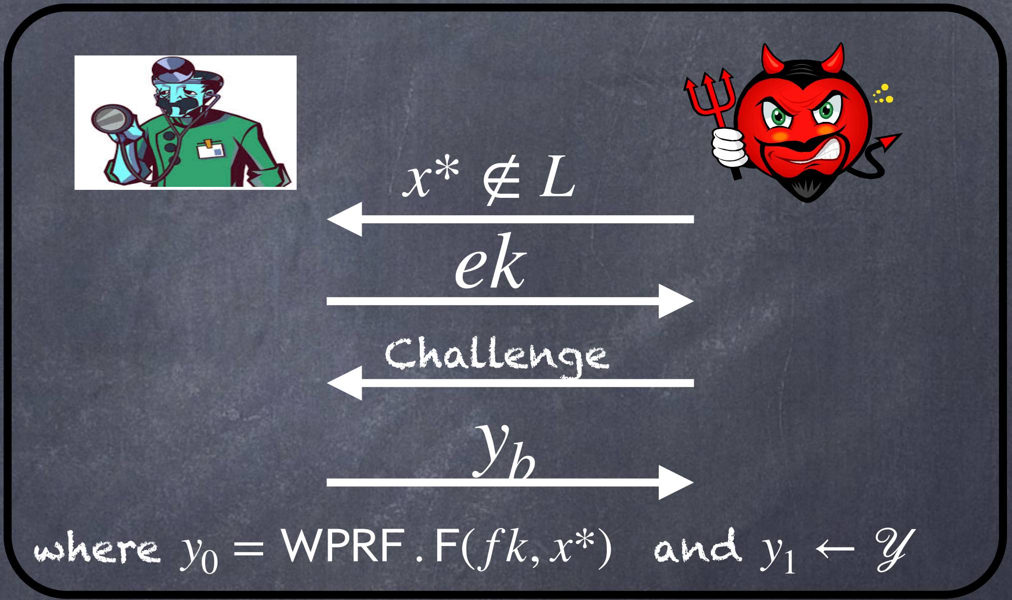
where $y_0 = \mathsf{WPRF.F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

(Non-reusable)DV-UWM => (NI)-WPRF

Intuition for Proof of (static-instance) Non-Interactive Security:

Recall $w^* \leftarrow \mathsf{dv.map}(crs, \hat{x}, (\perp, z))$ and $b = GL(w^*)$

- Convert a distinguisher for (NI)-WPRF to a predictor that outputs w^* w.h.p. s.t.
  - $\mathsf{dv.check}(crs, VK, \hat{x}, w^*) = 1$
- Switch $y \leftarrow \{0,1\}^{2\lambda}$ instead of $y = G(z)$.
- At this point the statement $\hat{x}$ is false; yet w^* is a valid proof for $\hat{x}$!



where $y_0 = \mathsf{WPRF.F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

$x^* \notin L$ and y is random

Contradiction!

(NI)-WPRF => Interactive WPRF

- $\mathsf{WPRF.Gen}(\lambda)$:

- $\mathsf{WPRF.F}(fk, x)$:

- $\mathsf{WPRF.Eval}(ek, x, w)$:

(NI)-WPRF => Interactive WPRF

- WPRF . Gen(λ):

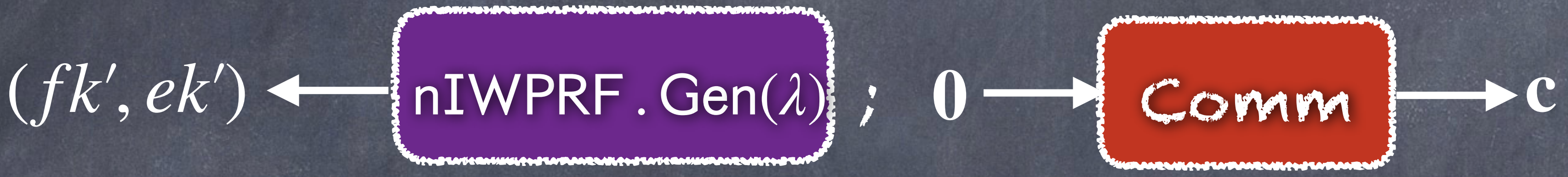
  $(fk', ek') \leftarrow \boxed{\text{nIWPRF . Gen}(\lambda)}$

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(NI)-WPRF => Interactive WPRF

- WPRF . Gen(λ):



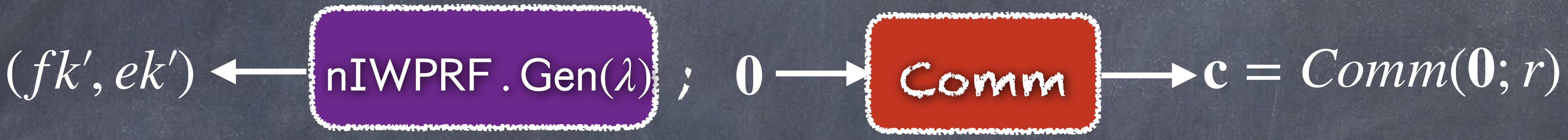
(fk', ek') ← nIWPRF . Gen(λ) ; 0 → Comm → c

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(NI)-WPRF => Interactive WPRF

- WPRF . Gen(λ):



$(fk', ek') \leftarrow$ nIWPRF . Gen(λ) ; $\mathbf{0} \rightarrow$ Comm $\rightarrow \mathbf{c} = Comm(\mathbf{0}; r)$

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(NI)-WPRF => Interactive WPRF

- WPRF . Gen(λ):

Statistically binding

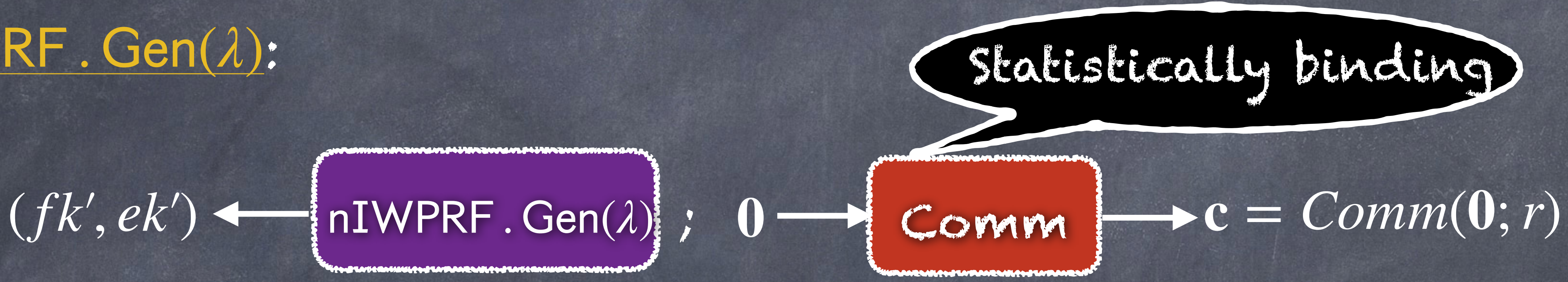
$$(fk', ek') \leftarrow \boxed{\text{nIWPRF . Gen}(\lambda)} \;;\; \mathbf{0} \rightarrow \boxed{\text{Comm}} \rightarrow \mathbf{c} = Comm(\mathbf{0}; r)$$

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(NI)-WPRF => Interactive WPRF

- WPRF . Gen(λ):



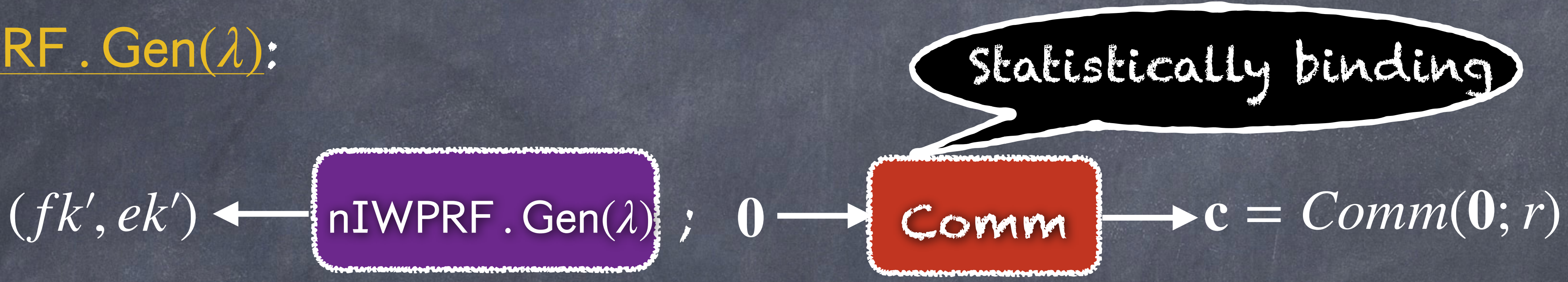
Set $ek = (ek', c)$ and $fk = (fk', r)$

- WPRF . F(fk, x):

- WPRF . Eval(ek, x, w):

(NI)-WPRF => Interactive WPRF

- WPRF . Gen(λ):



Statistically binding

$(fk', ek') \leftarrow$ nIWPRF . Gen(λ) ; $\mathbf{0} \rightarrow$ Comm $\rightarrow \mathbf{c} = Comm(\mathbf{0}; r)$

Set $ek = (ek', c)$ and $fk = (fk', r)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR $c = Comm(\tilde{x})$, where $\tilde{x} \neq x$"

- WPRF . Eval(ek, x, w):

(NI)-WPRF => Interactive WPRF

- WPRF . Gen(λ):

Statistically binding

$(fk', ek') \longleftarrow \boxed{\text{nIWPRF . Gen}(\lambda)}$; $\mathbf{0} \longrightarrow \boxed{\text{Comm}} \longrightarrow \mathbf{c} = Comm(\mathbf{0}; r)$

Set $ek = (ek', c)$ and $fk = (fk', r)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR $c = Comm(\tilde{x})$, where $\tilde{x} \neq x$"

$\mathbf{0} \longrightarrow \boxed{\text{Comm}} \longrightarrow \mathbf{c} = Comm(\mathbf{0}; r)$; $y \longleftarrow \boxed{\text{nIWPRF . F}(fk', (x, c))}$

- WPRF . Eval(ek, x, w):

(NI)-WPRF => Interactive WPRF

- WPRF . Gen(λ):

Statistically binding

$$(fk', ek') \leftarrow \boxed{\text{nIWPRF . Gen}(\lambda)} \;;\quad \mathbf{0} \rightarrow \boxed{\text{Comm}} \rightarrow \mathbf{c} = Comm(\mathbf{0}; r)$$

Set $ek = (ek', c)$ and $fk = (fk', r)$

- WPRF . F(fk, x):

Consider " $\hat{x} = x \in L$ OR $c = Comm(\tilde{x})$, where $\tilde{x} \neq x$"

$$\mathbf{0} \rightarrow \boxed{\text{Comm}} \rightarrow \mathbf{c} = Comm(\mathbf{0}; r) \;; \qquad y \leftarrow \boxed{\text{nIWPRF . F}(fk', (x, c))}$$

- WPRF . Eval(ek, x, w):

$$y \leftarrow \boxed{\text{nIWPRF . Eval}(ek', (x, c), (w, \perp, \perp))}$$

(NI)-WPRF => Interactive WPRF

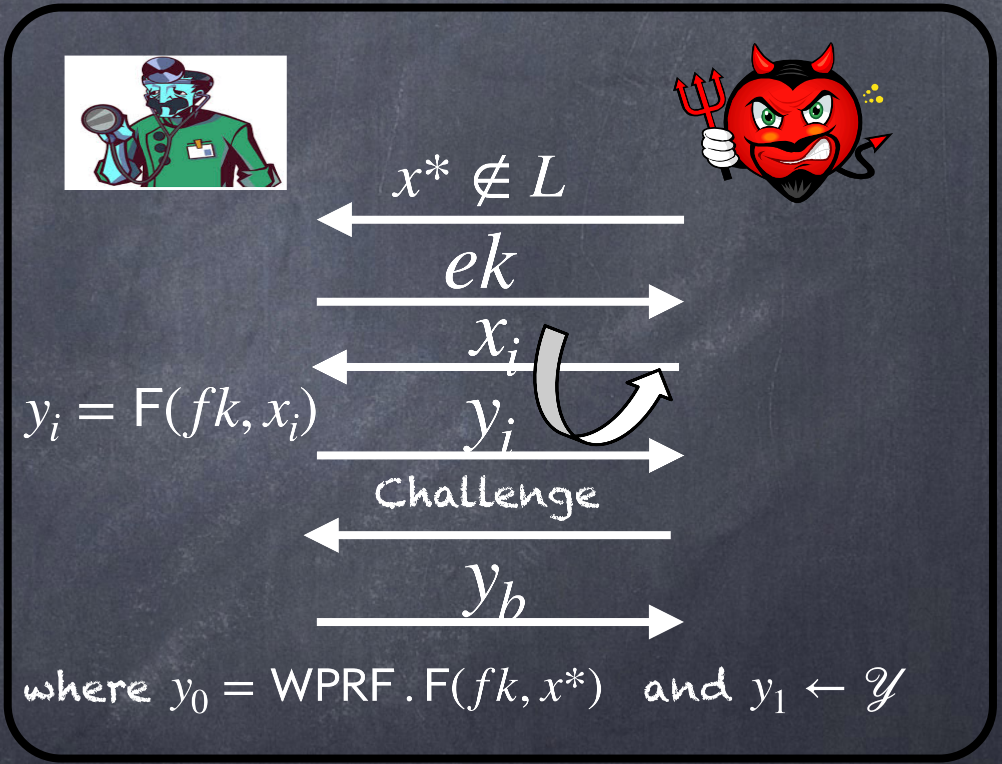
Intuition for Proof of (static-instance) Interactive Security:

Recall $y \leftarrow \mathsf{nIWPRF}.\mathsf{F}(fk', (x, c))$; where $c = Comm(\mathbf{0}; r)$

(NI)-WPRF => Interactive WPRF

Intuition for Proof of (static-instance) Interactive Security:

Recall $y \leftarrow \mathsf{nIWPRF.F}(fk', (x, c))$; where $c = Comm(\mathbf{0}; r)$



$x^* \notin L$

ek

x_i

$y_i = \mathsf{F}(fk, x_i)$ y_i

Challenge

y_b

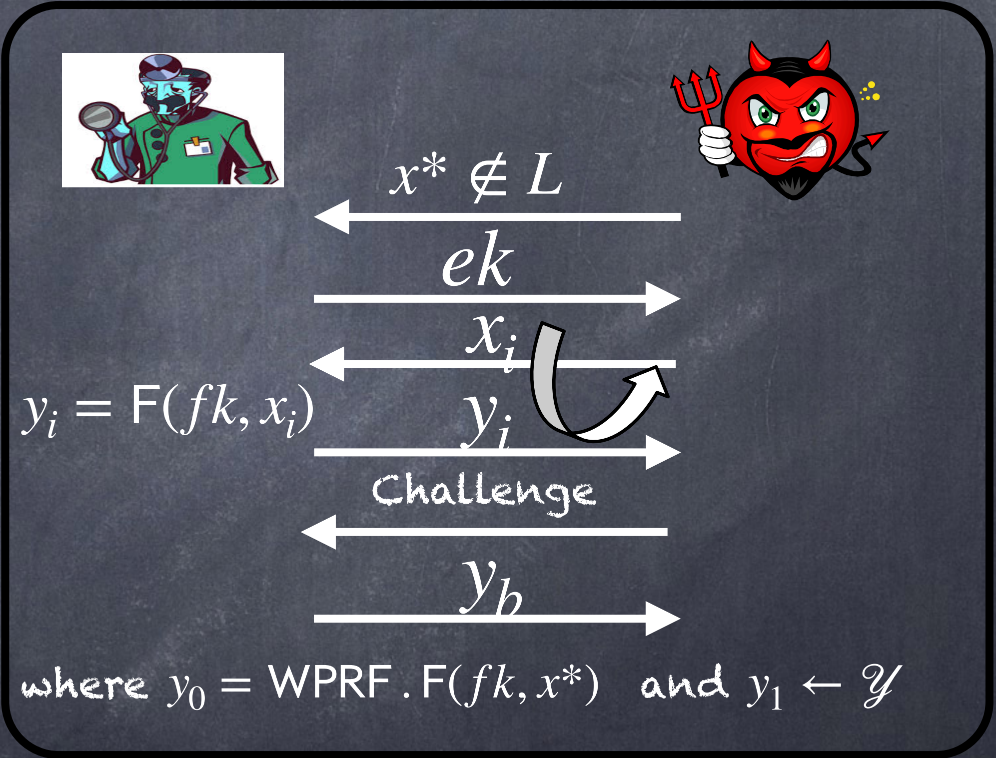
where $y_0 = \mathsf{WPRF.F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

(NI)-WPRF => Interactive WPRF

Intuition for Proof of (static-instance) Interactive Security:

Recall $y \leftarrow \mathsf{nIWPRF.F}(fk', (x, c))$; where $c = Comm(\mathbf{0}; r)$

- **Game 0:** Original game



$x^* \notin L$

ek

x_i

$y_i = \mathsf{F}(fk, x_i)$ y_i

Challenge

y_b

where $y_0 = \mathsf{WPRF.F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

(NI)-WPRF => Interactive WPRF

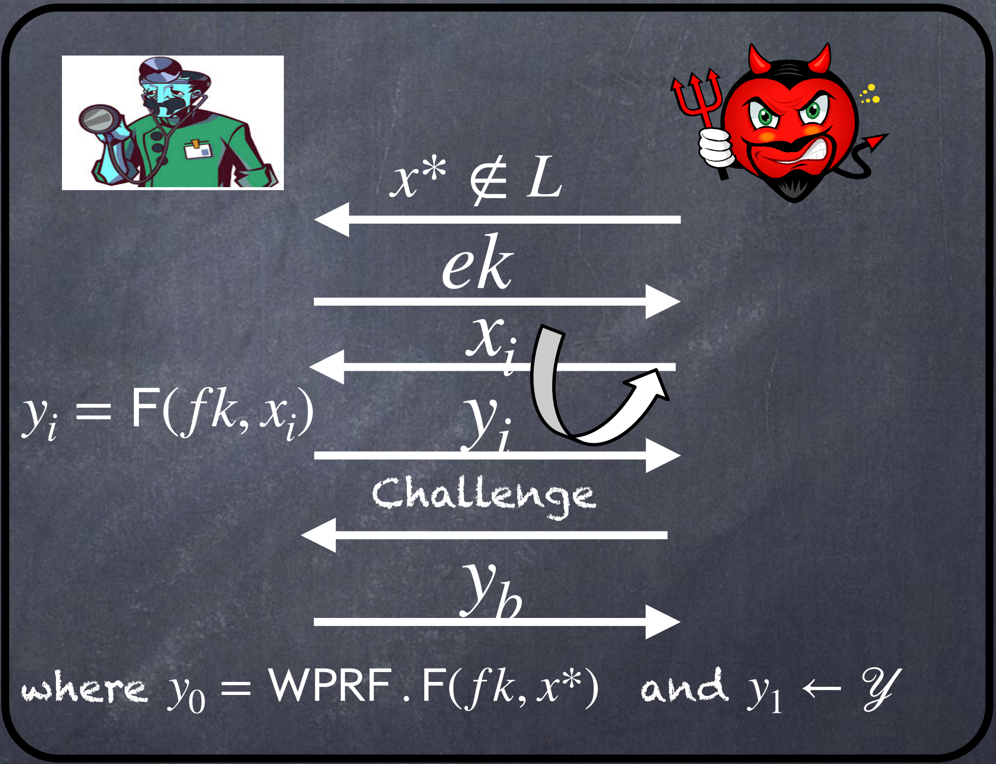
Intuition for Proof of (static-instance) Interactive Security:

Recall $y \leftarrow \mathsf{nIWPRF}.\mathsf{F}(fk', (x, c))$; where $c = Comm(\mathbf{0}; r)$

- **Game 0**: Original game

- **Game 1**: Compute $c^* = Comm(\mathbf{x}^*; r^*)$;

instead of $c^* = Comm(\mathbf{0}; r)$.



$x^* \notin L$

ek

x_i

$y_i = \mathsf{F}(fk, x_i)$ y_i

Challenge

y_b

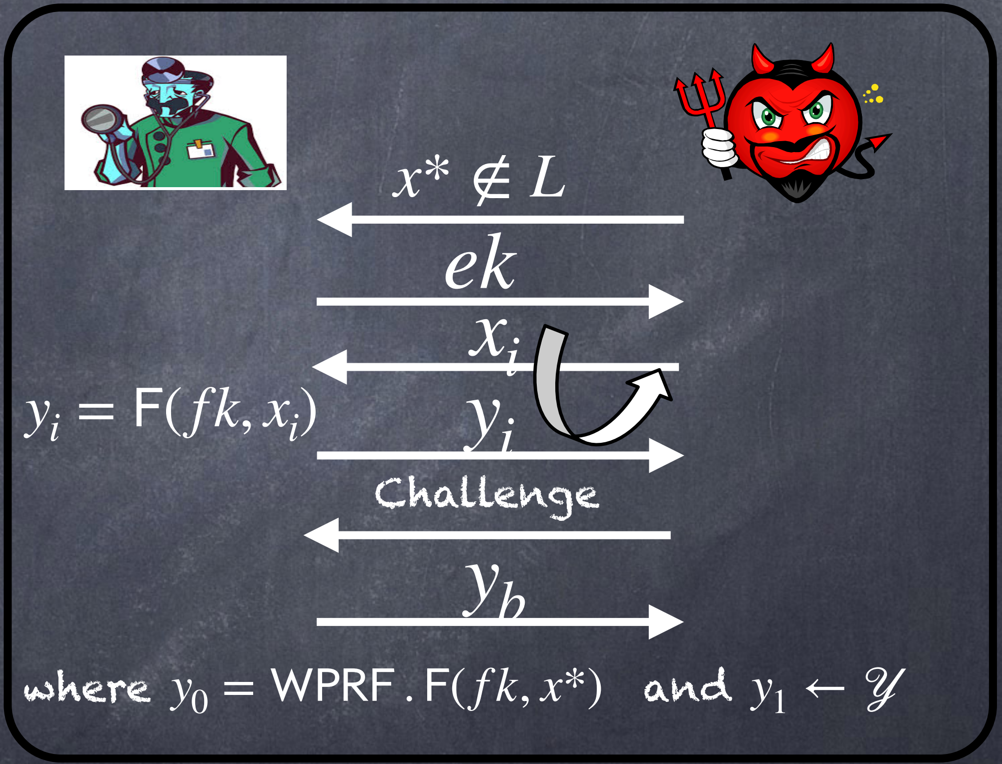
where $y_0 = \mathsf{WPRF}.\mathsf{F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

(NI)-WPRF => Interactive WPRF

Intuition for Proof of (static-instance) Interactive Security:

Recall $y \leftarrow \mathsf{nIWPRF.F}(fk', (x, c))$; where $c = Comm(\mathbf{0}; r)$

- **Game 0**: Original game
- **Game 1**: Compute $c^* = Comm(\mathbf{x}^*; r^*)$; instead of $c^* = Comm(\mathbf{0}; r)$.
  - Evaluation queries x_i are answered using $y \leftarrow \mathsf{nIWPRF.F}(fk', (x_i, c^*))$



where $y_0 = \mathsf{WPRF.F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

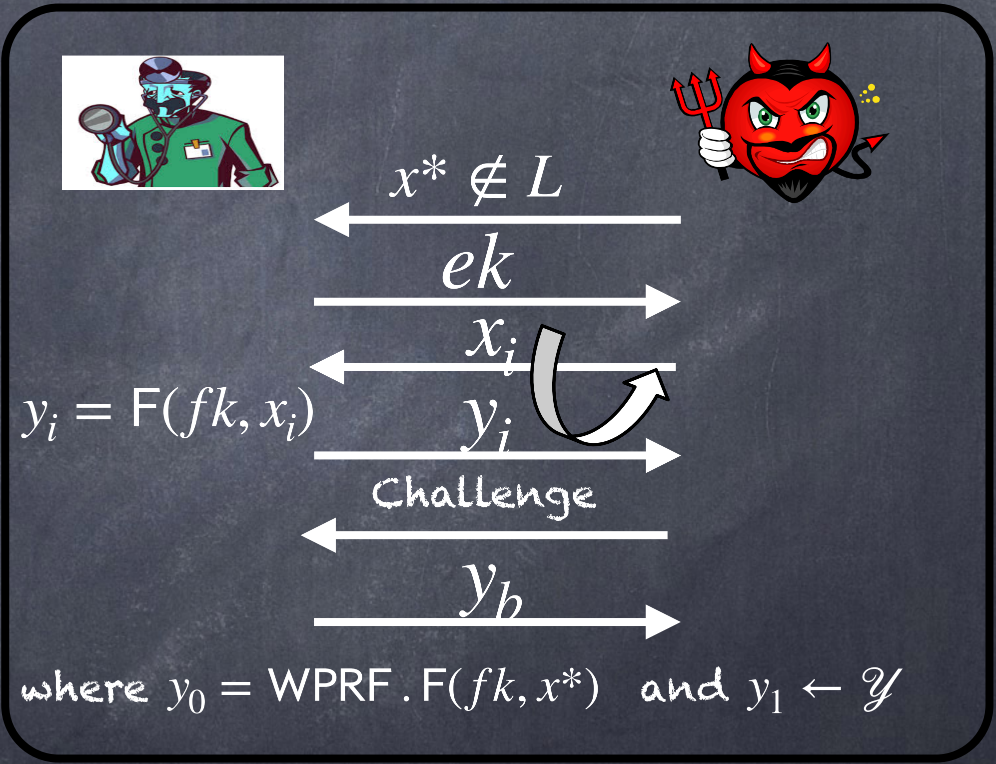
(NI)-WPRF => Interactive WPRF

Intuition for Proof of (static-instance) Interactive Security:

Recall $y \leftarrow \mathsf{nIWPRF.F}(fk', (x, c))$; where $c = Comm(\mathbf{0}; r)$

- **Game 0**: Original game
- **Game 1**: Compute $c^* = Comm(\mathbf{x}^*; r^*)$; instead of $c^* = Com$ [illegible]
  - Evaluation queries x_i [illegible] answered using $y \leftarrow \mathsf{nIWPRF.F}(fk', (x_i, c^*))$

$(x_i, c^*) \in L$



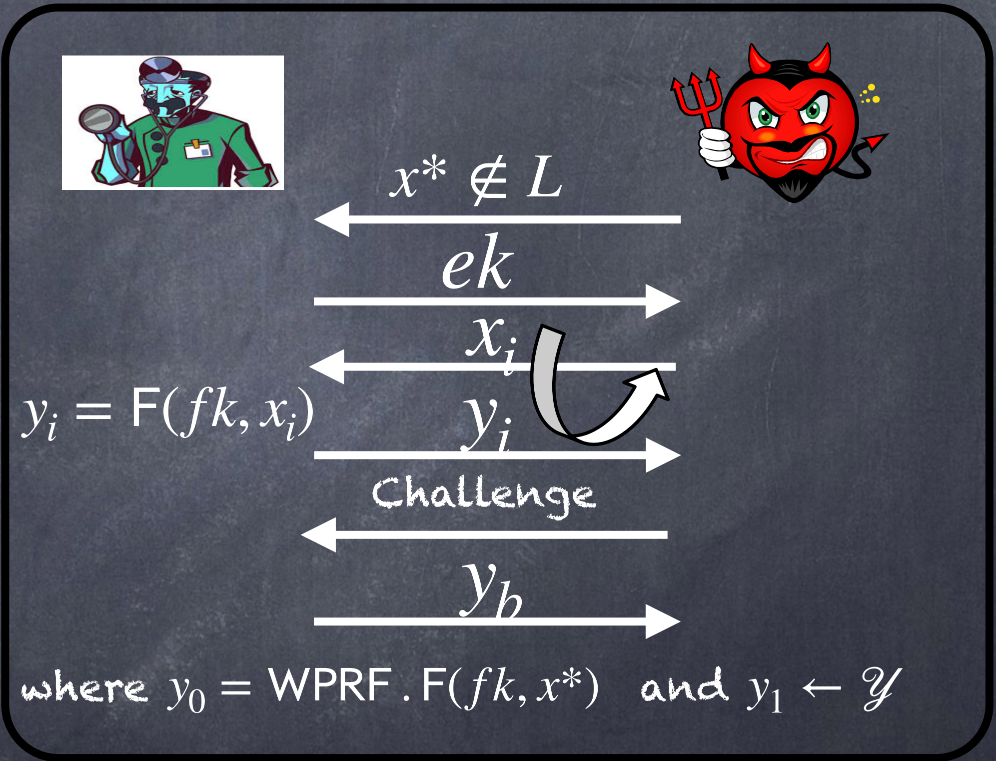
(NI)-WPRF => Interactive WPRF

Intuition for Proof of (static-instance) Interactive Security:

Recall $y \leftarrow \mathsf{nIWPRF.F}(fk', (x, c))$; where $c = Comm(\mathbf{0}; r)$

- **Game 0**: Original game

- **Game 1**: Compute $c^* = Comm(\mathbf{x}^*; r^*)$; instead of $c^* = Comm(\mathbf{0}; r)$.
    - Evaluation queries x_i are answered using $y \leftarrow \mathsf{nIWPRF.F}(fk', (x_i, c^*))$



$x^* \notin L$

ek

x_i

$y_i = \mathsf{F}(fk, x_i)$

y_i

Challenge

y_b

where $y_0 = \mathsf{WPRF.F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

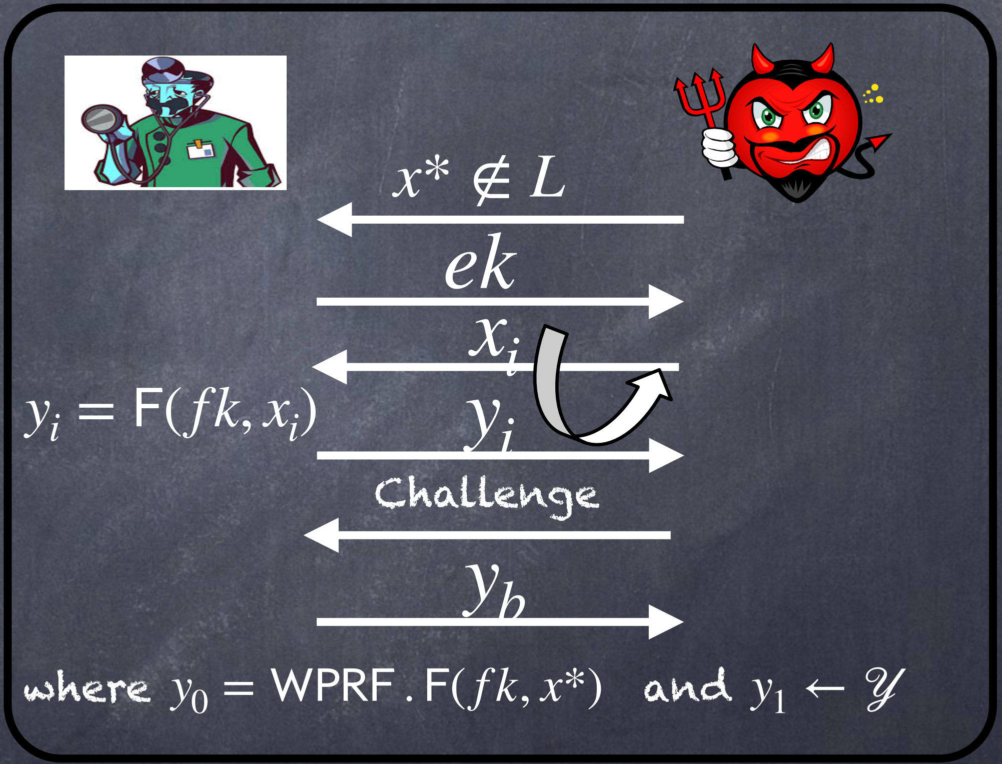
- **Game 2**: Evaluation queries x_i are answered using $y \leftarrow \mathsf{nIWPRF.Eval}(ek', (x_i, c^*), (\perp, x^*, r^*))$

(NI)-WPRF => Interactive WPRF

Intuition for Proof of (static-instance) Interactive Security:

Recall $y \leftarrow \mathsf{nIWPRF}.\mathsf{F}(fk', (x, c))$; where $c = Comm(\mathbf{0}; r)$

- **Game 0**: Original game



$x^* \notin L$

ek

x_i

$y_i = \mathsf{F}(fk, x_i)$

y_i

Challenge

y_b

where $y_0 = \mathsf{WPRF}.\mathsf{F}(fk, x^*)$ and $y_1 \leftarrow \mathcal{Y}$

- **Game 1**: Compute $c^* = Comm(\mathbf{x}^*; r^*)$; instead of $c^* = Comm(\mathbf{0}; r)$.
  - Evaluation queries x_i are answered using $y \leftarrow \mathsf{nIWPRF}.\mathsf{F}(fk', (x_i, c^*))$
- **Game 2**: Evaluation queries x_i are answered using $y \leftarrow \mathsf{nIWPRF}.\mathsf{Eval}(ek', (x_i, c^*), (\perp, x^*, r^*))$
- Advantage of Adv. in Game 2 is negligible.

Open Problems

Open Problems

- Connection of witness maps with other crypto primitives?

Open Problems

- Connection of witness maps with other crypto primitives?
    - For e.g., does witness map imply SNARG?

Open Problems

- Connection of witness maps with other crypto primitives?
    - For e.g., does witness map imply SNARG?
    - If not, is it possible to black-box separate witness maps from falsifiable assumptions?

Open Problems

- Connection of witness maps with other crypto primitives?
  - For e.g., does witness map imply SNARG?
  - If not, is it possible to black-box separate witness maps from falsifiable assumptions?
- Does witness map imply OWF (under some complexity theoretic assumptions)?

Open Problems

- Connection of witness maps with other crypto primitives?
  - For e.g., does witness map imply SNARG?
  - If not, is it possible to black-box separate witness maps from falsifiable assumptions?
- Does witness map imply OWF (under some complexity theoretic assumptions)?
- More applications of witness maps?

THAT'S ALL FOLKS!

THANK YOU :-)

https://eprint.iacr.org/2023/343