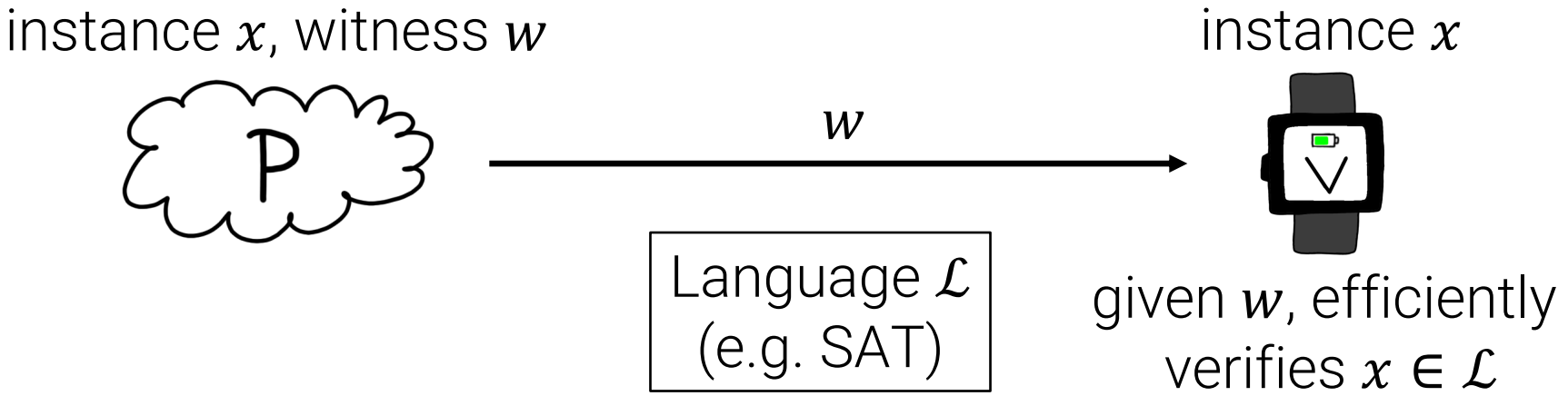
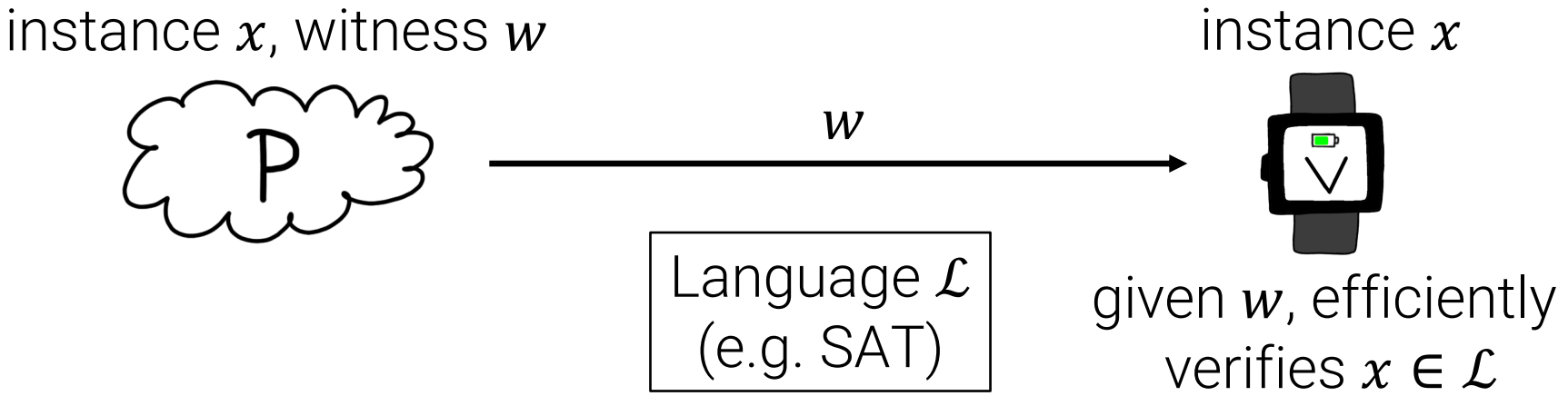
On the Insecurity of Kilian-Based SNARGs

James Bartusek (UC Berkeley)
Liron Bronfman (Technion)
Justin Holmgren (Simons Institute)
Fermi Ma (Princeton)
Ron D. Rothblum (Technion)

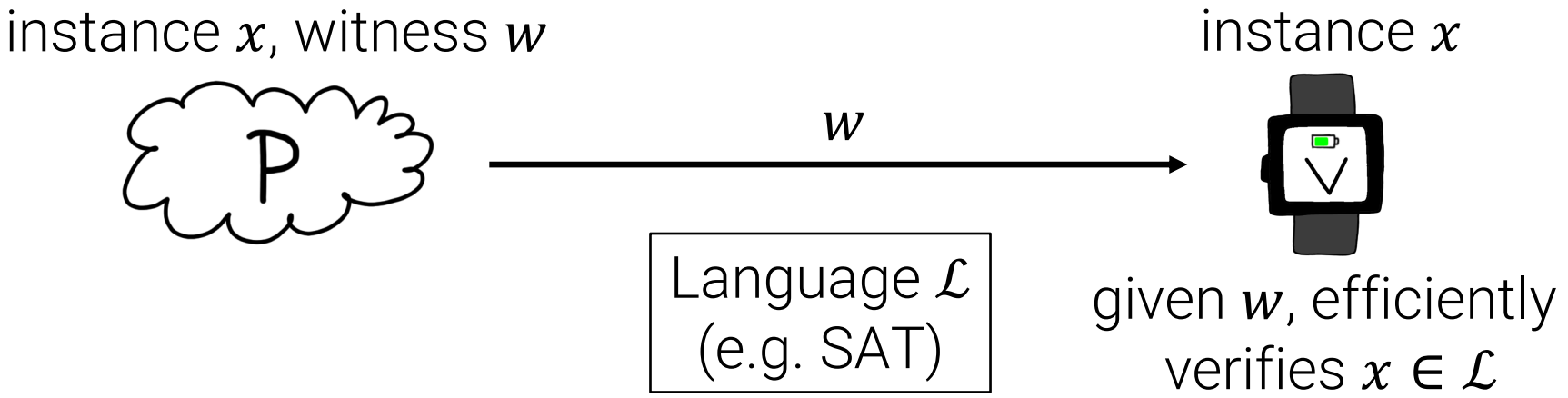


$\mathcal{L} \in NP$ means efficiently verifiable proofs



$\mathcal{L} \in NP$ means efficiently verifiable proofs

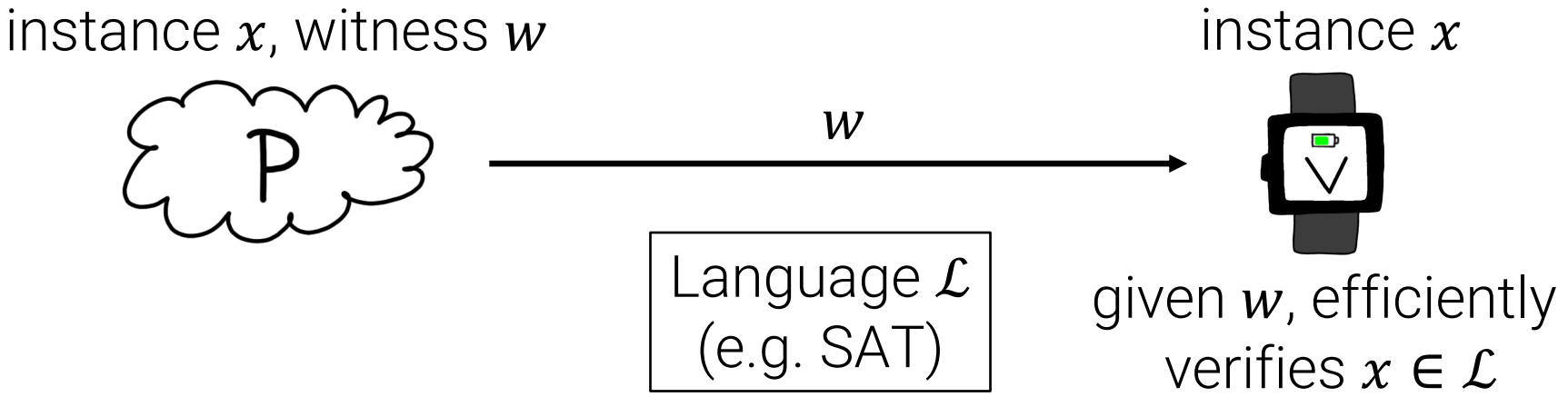
how efficient?



$\mathcal{L} \in NP$ means efficiently verifiable proofs

how efficient?

- w can't be too short: if $|w| = o(|x|)$, then $NP \in DTIME(2^{o(|x|)})$



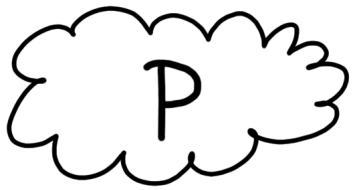
$\mathcal{L} \in NP$ means efficiently verifiable proofs

how efficient?

- w can't be too short: if $|w| = o(|x|)$, then $NP \in DTIME(2^{o(|x|)})$
- even with interaction [Goldreich-Håstad 98, Goldreich-Vadhan-Wigderson 02, Wee 05]

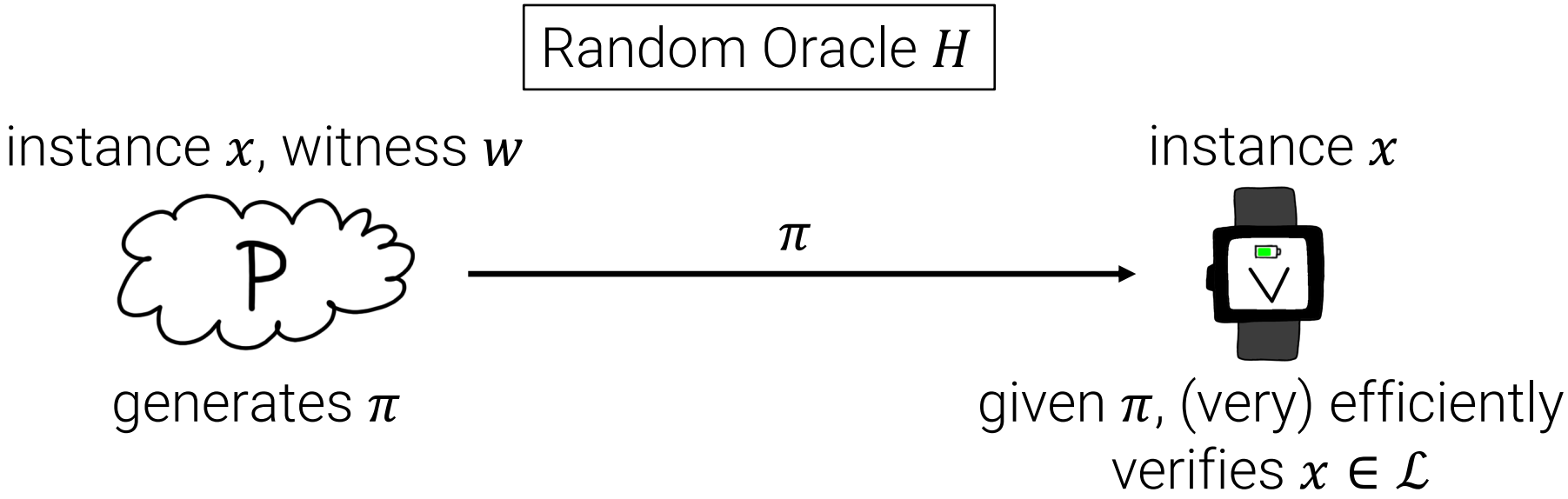
Random Oracle H

instance x, witness w



instance x

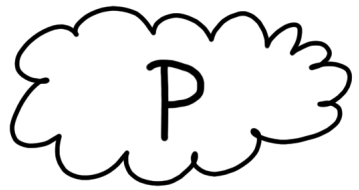
[Kilian 92, Micali 94]: Communication can be significantly shorter if we only require soundness against ***computationally efficient*** cheating 



[Kilian 92, Micali 94]: Communication can be significantly shorter if we only require soundness against ***computationally efficient*** cheating

Random Oracle H

instance x, witness w



generates π

π

instance x

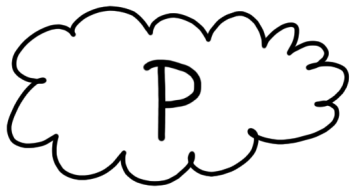
given π, (very) efficiently verifies $x \in \mathcal{L}$

[Kilian 92, Micali 94]: Communication can be significantly shorter if we only require soundness against ***computationally efficient*** cheating 

If  accepts, then either $x \in \mathcal{L}$ OR  inefficient

Random Oracle H

instance x, witness w



generates π

π

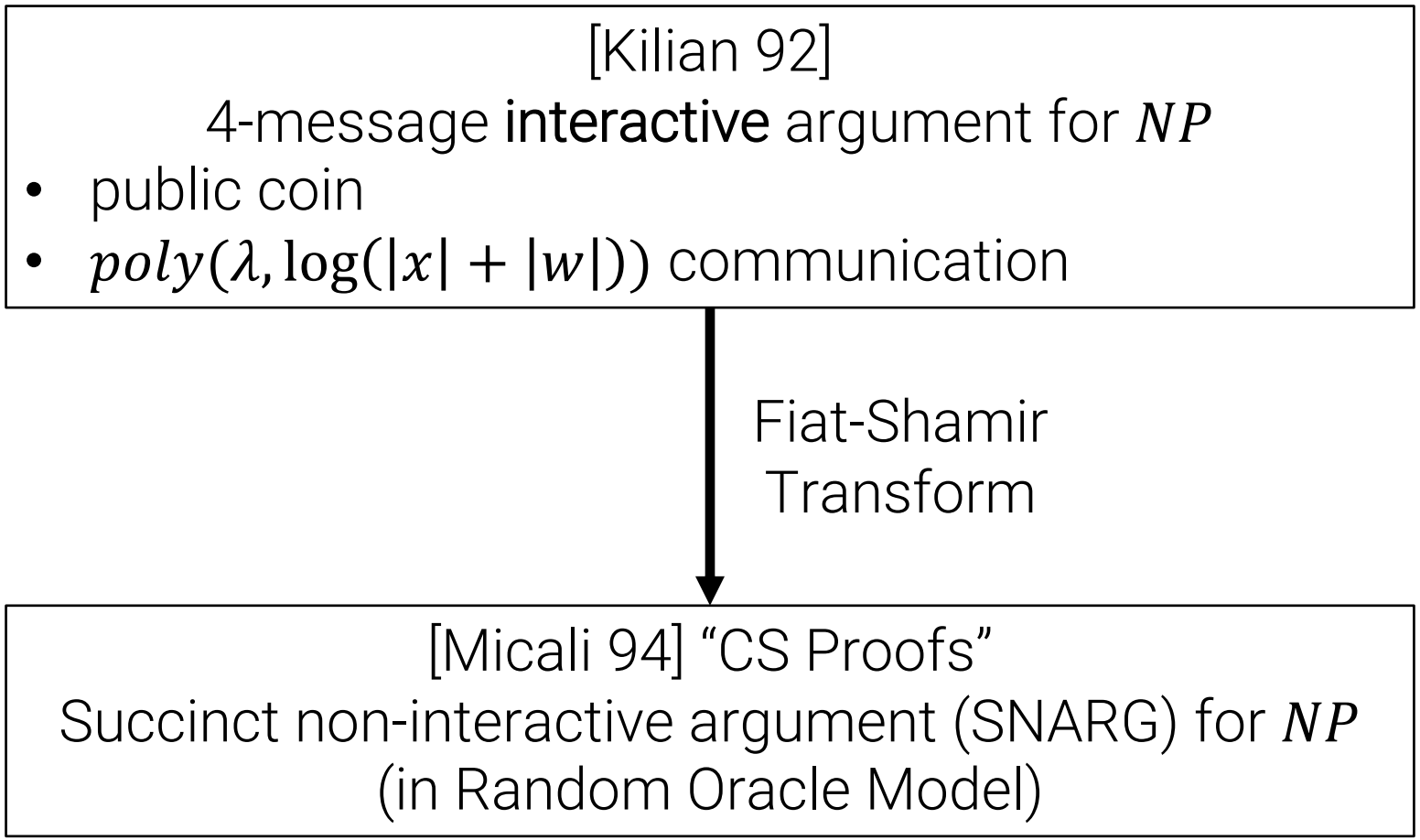
instance x

given π, (very) efficiently verifies $x \in \mathcal{L}$

[Kilian 92, Micali 94]: Communication can be significantly shorter if we only require soundness against ***computationally efficient*** cheating 

If  accepts, then either $x \in \mathcal{L}$ OR  inefficient

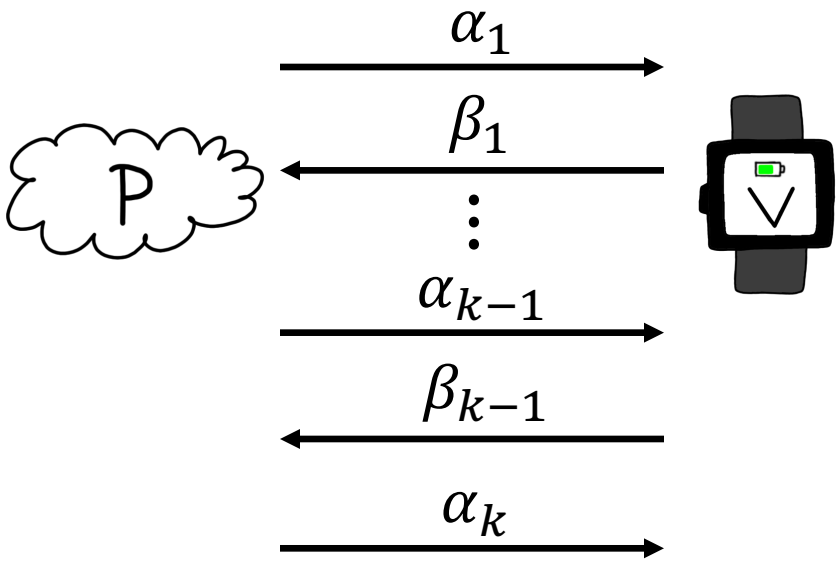
- "Succinct" means $|\pi| = poly(\lambda, \log(|x| + |w|))$
- "Argument" means sound against efficient cheating 


[Kilian 92]
4-message interactive argument for NP
• public coin
• $poly(\lambda, \log(|x| + |w|))$ communication
Fiat-Shamir
Transform
[Micali 94] "CS Proofs"
Succinct non-interactive argument (SNARG) for NP
(in Random Oracle Model)

The Fiat-Shamir Heuristic

[Fiat-Shamir 89]

public coin: β_i are uniformly random



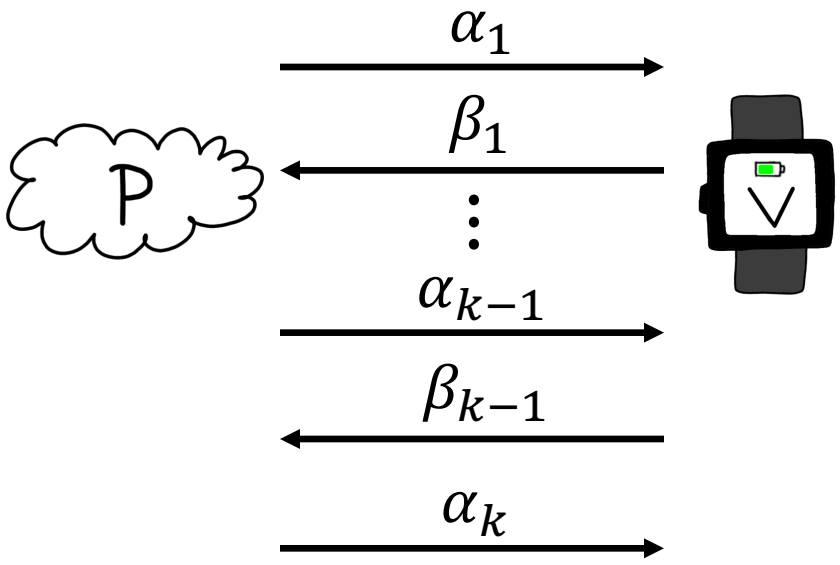
Interactive Protocol

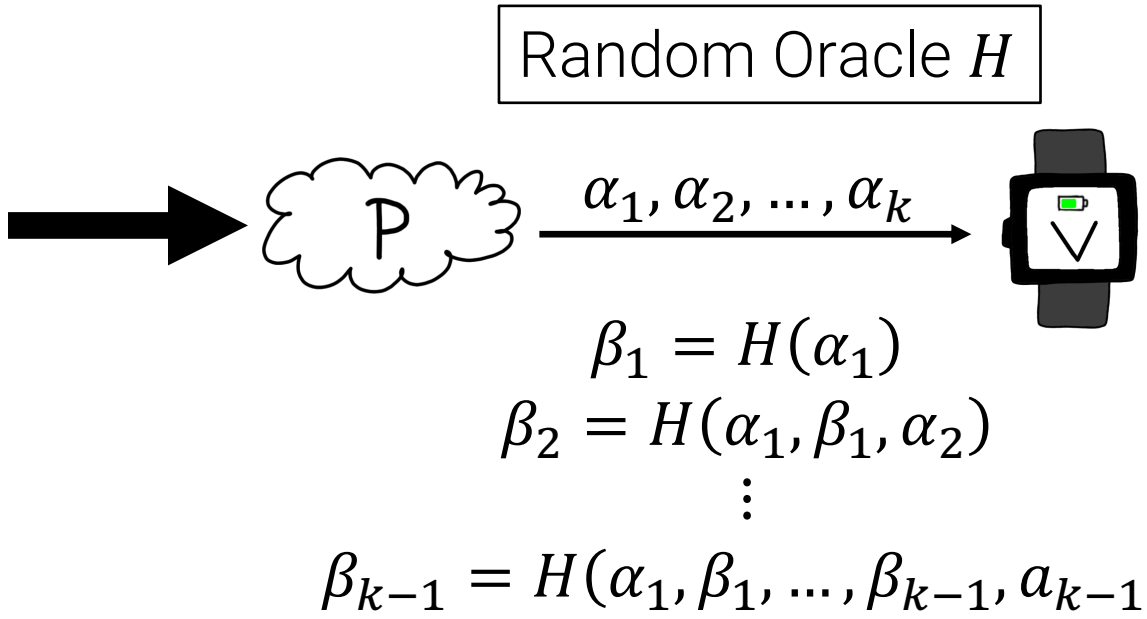
(ex: Kilian's 4-message argument for NP)

The Fiat-Shamir Heuristic

[Fiat-Shamir 89]

public coin: β_i are uniformly random





Interactive Protocol

Non-Interactive Argument

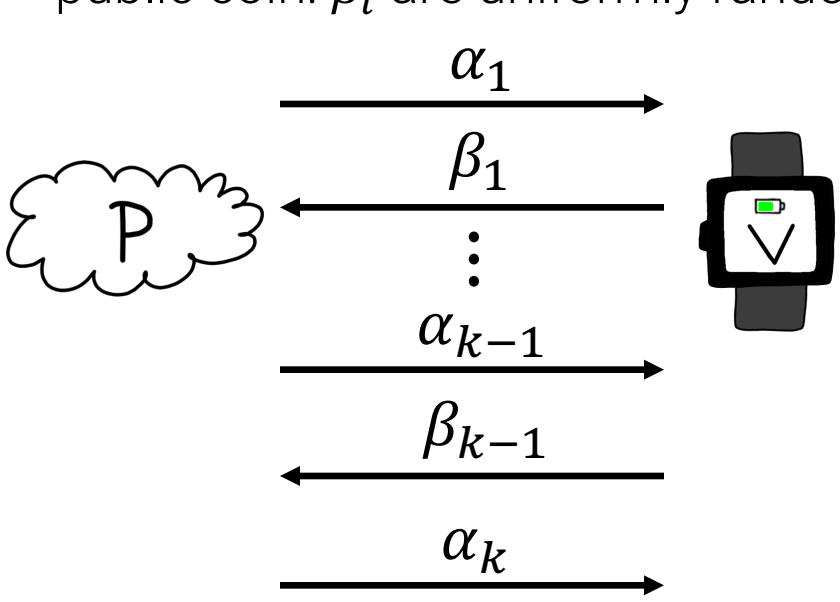
(ex: Kilian's 4-message argument for NP)

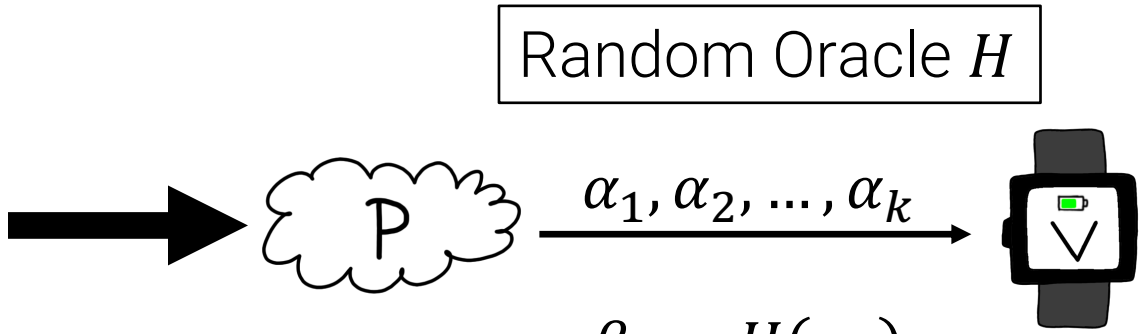
(ex: Micali's CS Proofs)

The Fiat-Shamir Heuristic

[Fiat-Shamir 89]

public coin: β_i are uniformly random





$$\beta_1 = H(\alpha_1)$$
$$\beta_2 = H(\alpha_1, \beta_1, \alpha_2)$$
$$\vdots$$
$$\beta_{k-1} = H(\alpha_1, \beta_1, \ldots, \beta_{k-1}, \alpha_{k-1})$$

Interactive Protocol

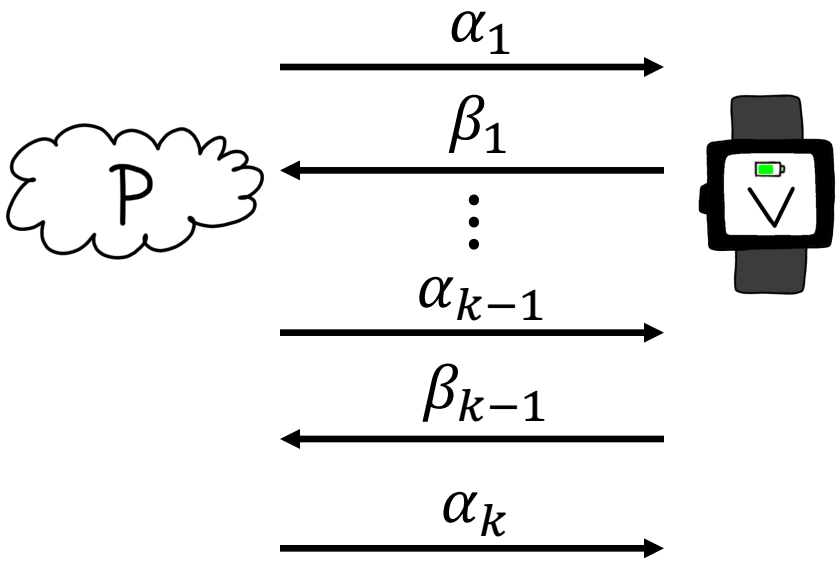
Non-Interactive Argument

[Pointcheval-Stern 96]:
Sound for constant-round arguments in ROM.

The Fiat-Shamir Heuristic

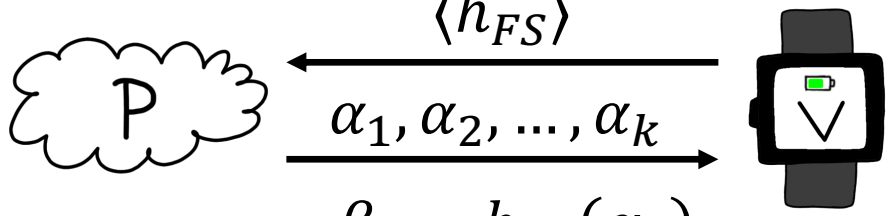
[Fiat-Shamir 89]

public coin: β_i are uniformly random



Interactive Protocol

What if we replace H with an explicit hash function?



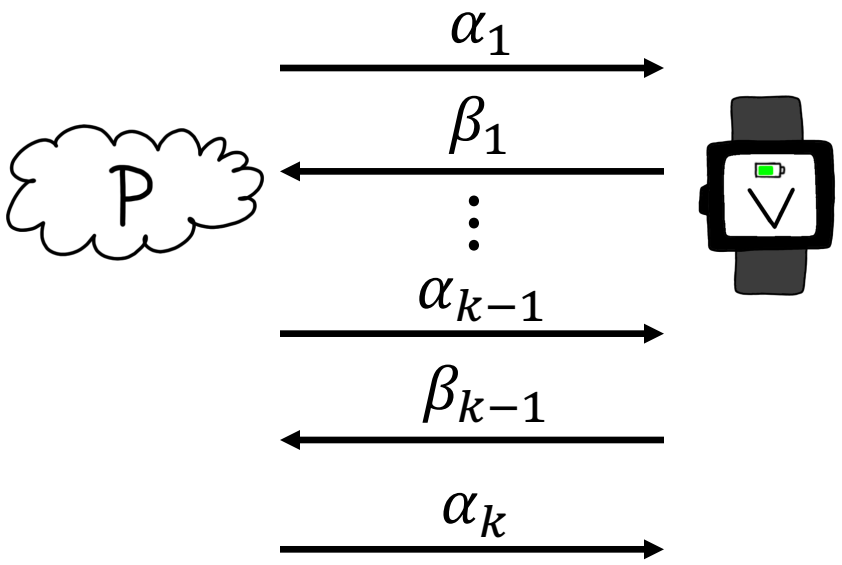
$$\beta_1 = h_{FS}(\alpha_1)$$
$$\beta_2 = h_{FS}(\alpha_1, \beta_1, \alpha_2)$$
$$\vdots$$
$$\beta_{k-1} = h_{FS}(\alpha_1, \beta_1, \ldots, \beta_{k-1}, \alpha_{k-1})$$

Non-Interactive Argument

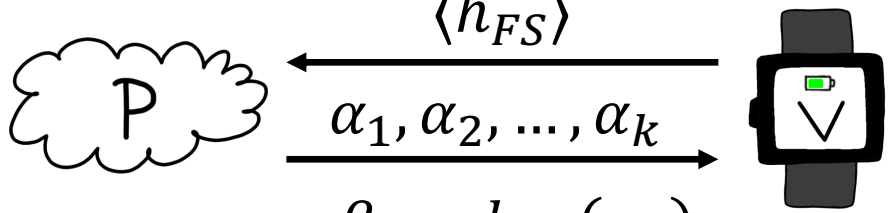
The Fiat-Shamir Heuristic

[Fiat-Shamir 89]

public coin: β_i are uniformly random



What if we replace H with an explicit hash function?

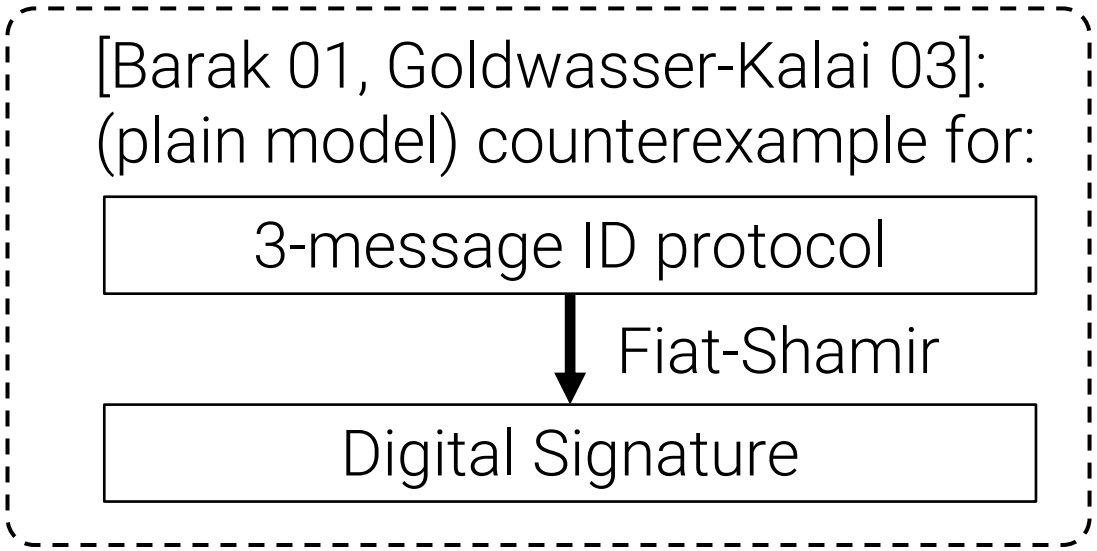


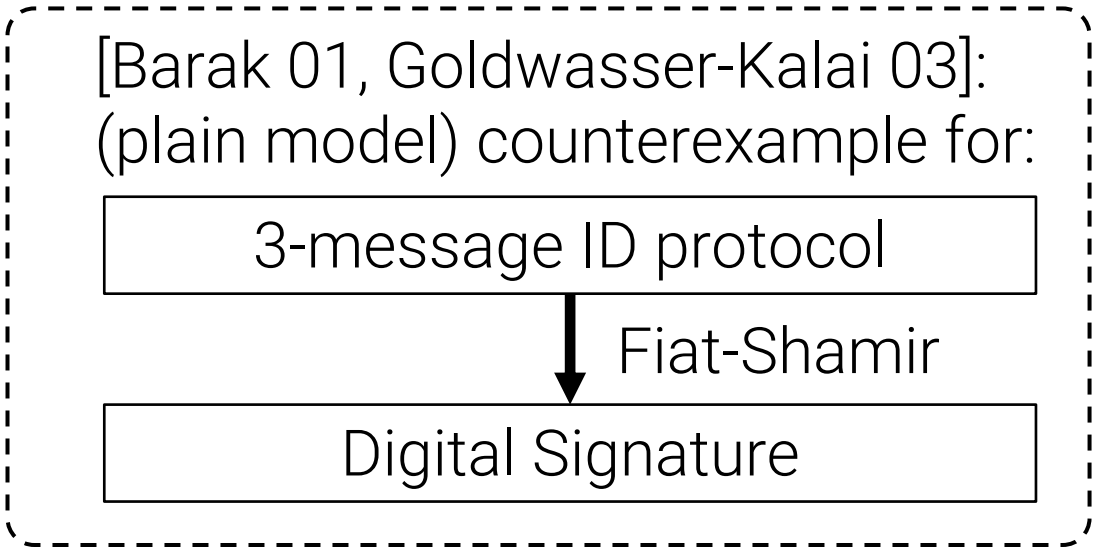
$$\beta_1 = h_{FS}(\alpha_1)$$
$$\beta_2 = h_{FS}(\alpha_1, \beta_1, \alpha_2)$$
$$\vdots$$
$$\beta_{k-1} = h_{FS}(\alpha_1, \beta_1, \ldots, \beta_{k-1}, a_{k-1})$$

Interactive Protocol

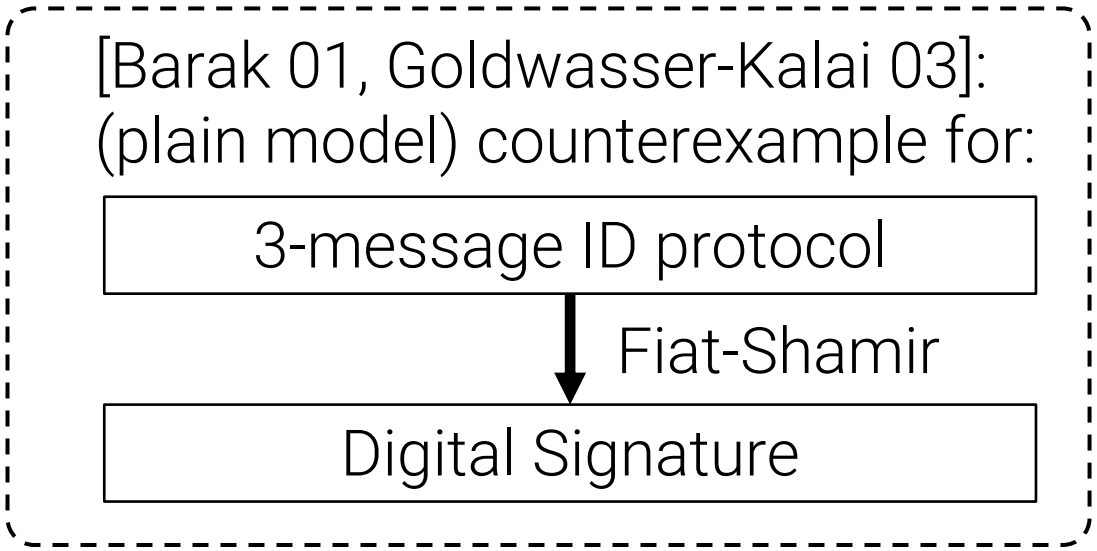
Non-Interactive Argument

[Barak 01, Goldwasser-Kalai 03]:
(Contrived) arguments where FS is unsound for any explicit h_{FS}.

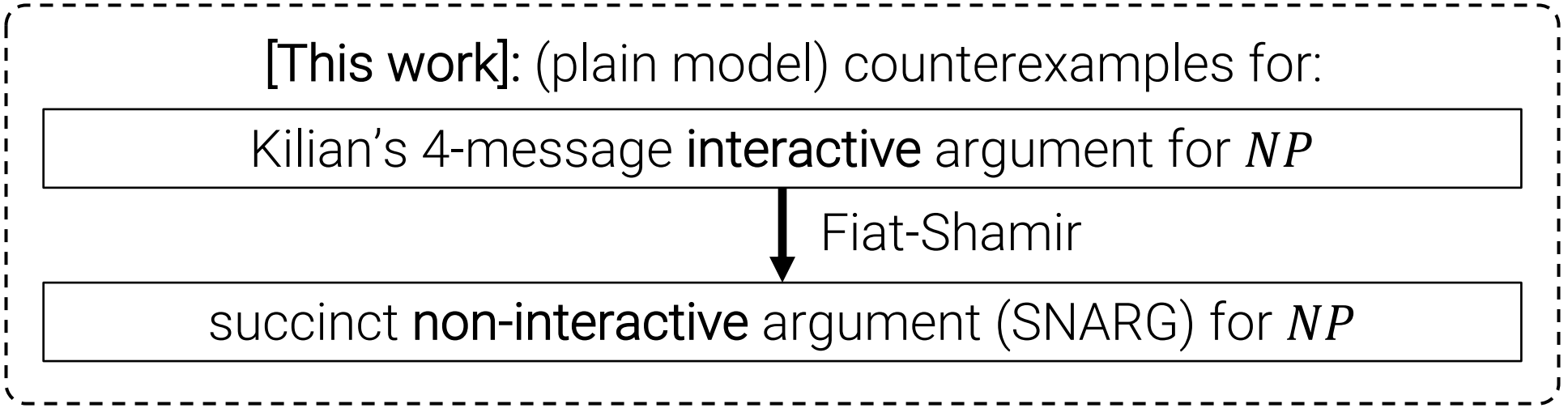

[Barak 01, Goldwasser-Kalai 03]:
(plain model) counterexample for:
3-message ID protocol
Fiat-Shamir
Digital Signature



Possible interpretation:
Counterexample is so contrived that “natural” cryptographic protocols (with more structure) should still be safe for Fiat-Shamir.



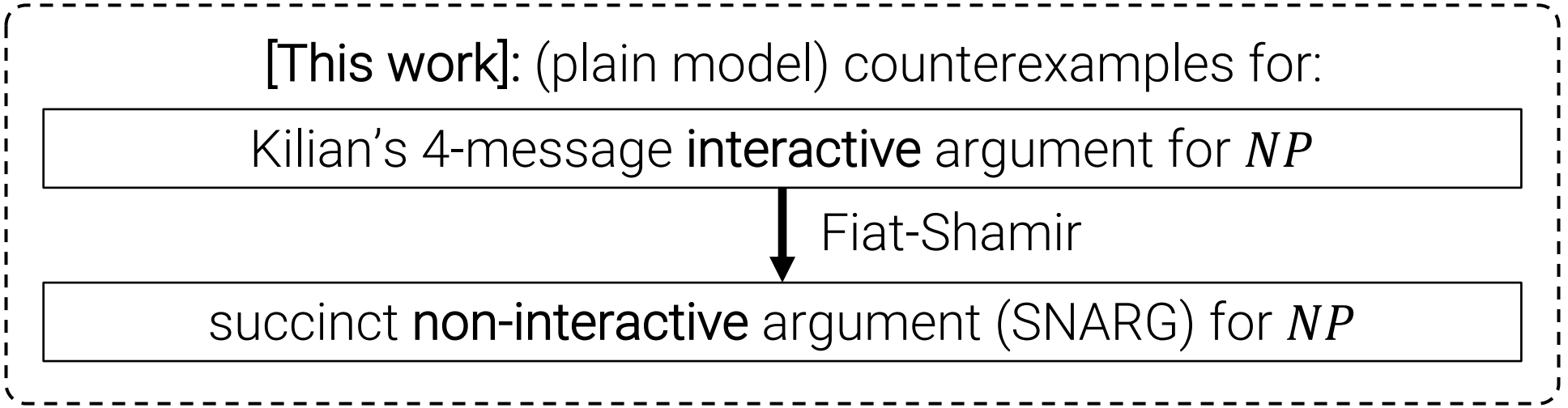
Possible interpretation:
Counterexample is so contrived that "natural" cryptographic protocols (with more structure) should still be safe for Fiat-Shamir.



(Kilian's protocol combines probabilistically checkable proof (PCP)/interactive oracle proof (IOP) with a CRHF.)

We show:

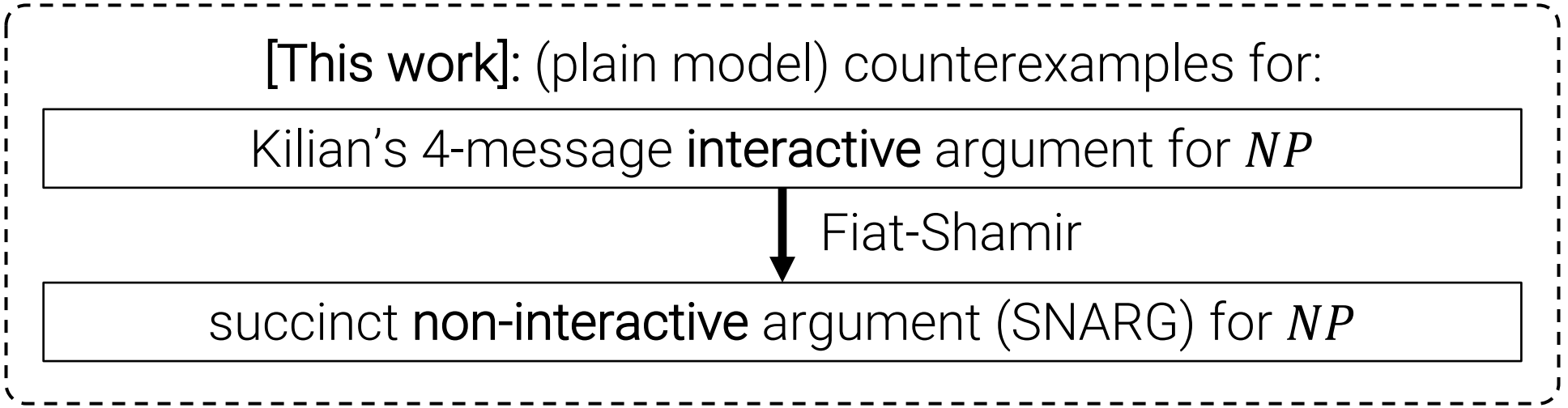
- IOP s.t. Fiat-Shamir + Kilian is always unsound



(Kilian's protocol combines probabilistically checkable proof (PCP)/interactive oracle proof (IOP) with a CRHF.)

We show:

- IOP s.t. Fiat-Shamir + Kilian is always unsound
- CRHF s.t. Fiat-Shamir + Kilian is (almost) always unsound



Fiat-Shamir-Kilian-Micali* used in some real-world SNARGs:

- SCI [Ben-Sasson, Ben-Tov, Chiesa, Gabizon, Genkin, Hamilis, Pergament, Riabzev, Silberstein, Tromer, Virza 17]
- Aurora [Ben-Sasson-Bentov-Horesh-Riabzev 18]
- STARK [Ben-Sasson-Chiesa-Riabzev-Spooner-Virza-Ward 19]

*applied to Kilian's protocol for interactive oracle proofs (IOPs)

Fiat-Shamir-Kilian-Micali* used in some real-world SNARGs:

- SCI [Ben-Sasson, Ben-Tov, Chiesa, Gabizon, Genkin, Hamilis, Pergament, Riabzev, Silberstein, Tromer, Virza 17]
- Aurora [Ben-Sasson-Bentov-Horesh-Riabzev 18]
- STARK [Ben-Sasson-Chiesa-Riabzev-Spooner-Virza-Ward 19]

+ only known approach to post-quantum publicly verifiable SNARGs

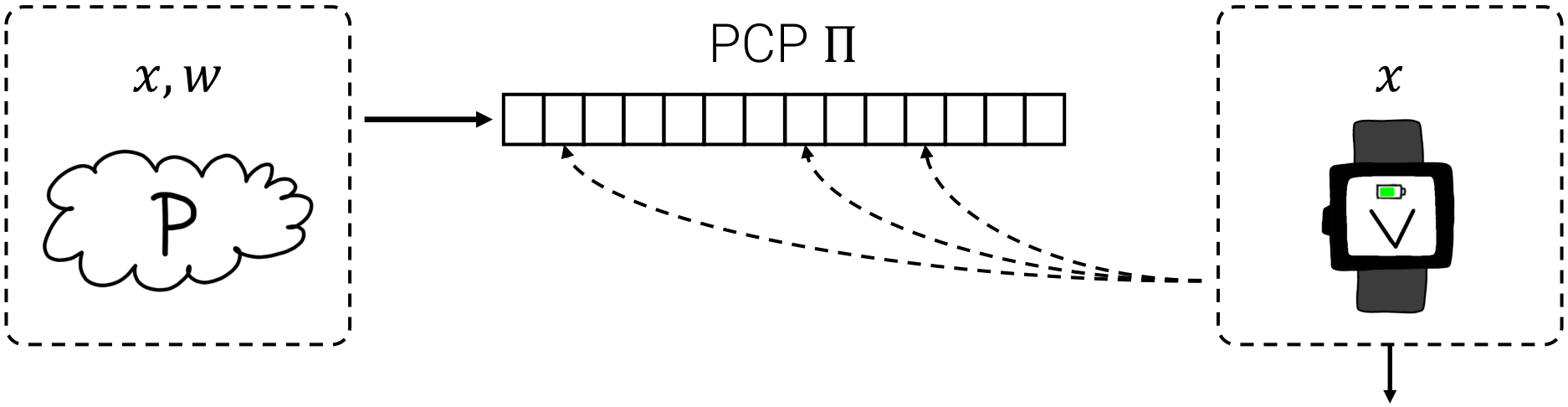
*applied to Kilian's protocol for interactive oracle proofs (IOPs)

Outline

- (Review) Kilian’s Protocol
- A Bad IOP for IOP-Based SNARGs
- (If time) A Bad CRHF for PCP-Based SNARGs

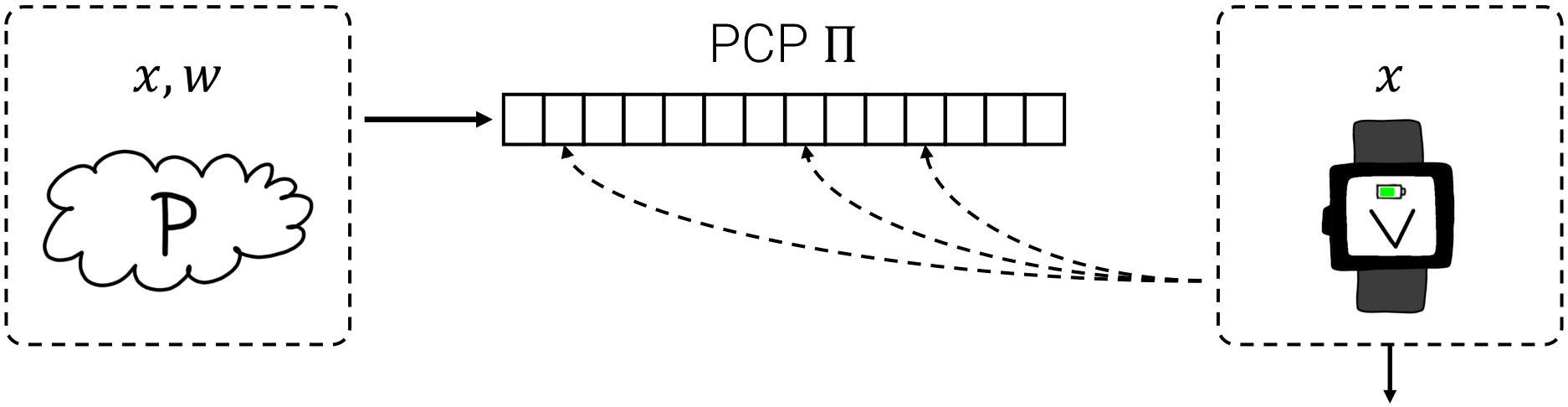
Probabilistically Checkable Proofs (PCPs)

[Fortnow-Rompel-Sipser 88, Feige-Goldwasser-Lovász-Safra-Szegedy 91, Babai-Fortnow-Levin-Szegedy 91, Arora-Safra 92, Arora-Lund-Mutwani-Sudan-Szegedy 92]



Probabilistically Checkable Proofs (PCPs)

[Fortnow-Rompel-Sipser 88, Feige-Goldwasser-Lovász-Safra-Szegedy 91, Babai-Fortnow-Levin-Szegedy 91, Arora-Safra 92, Arora-Lund-Mutwani-Sudan-Szegedy 92]



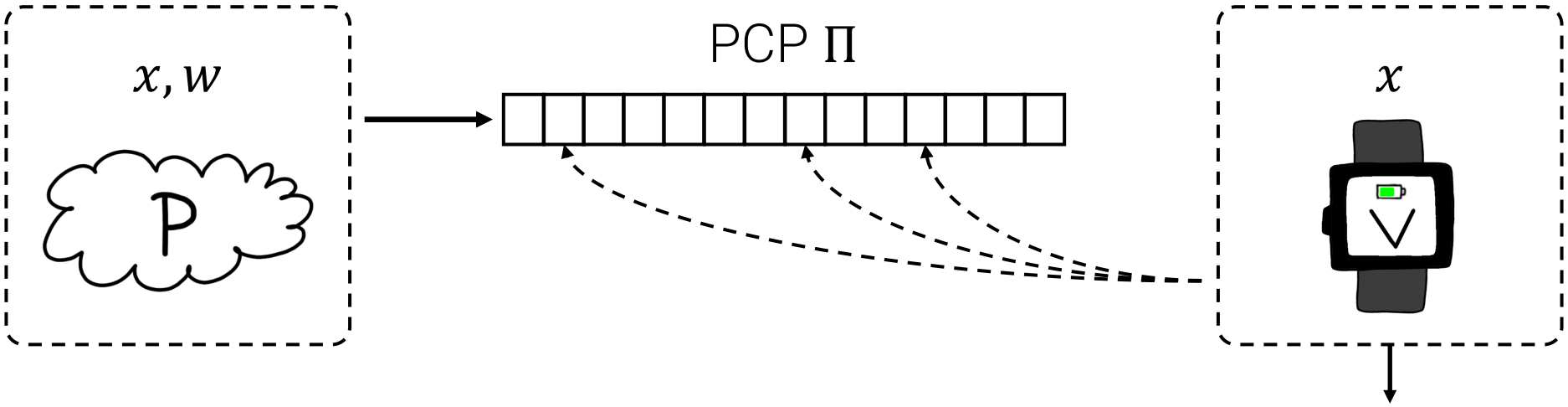
Claim: $x \in \mathcal{L}$

Thm. If $\mathcal{L} \in NP$, V making $O(1)$ queries:

- accepts if $x \in \mathcal{L}$
- rejects if $x \notin \mathcal{L}$ with $1/2$ probability.

Probabilistically Checkable Proofs (PCPs)

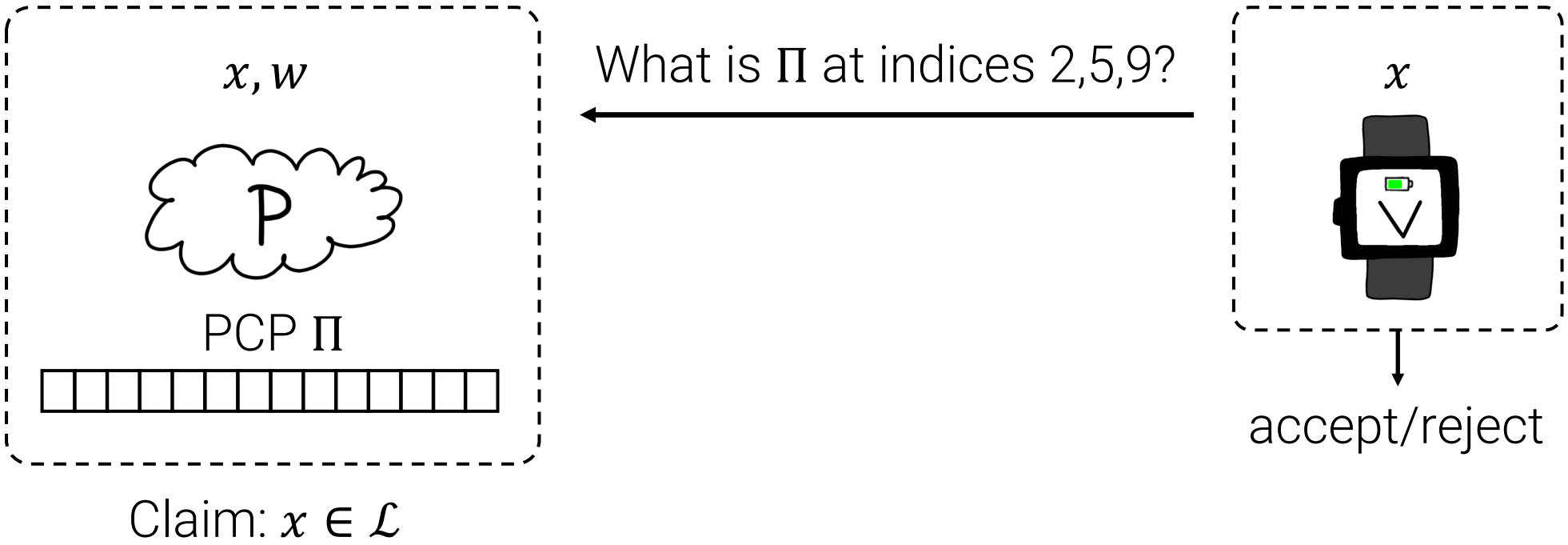
[Fortnow-Rompel-Sipser 88, Feige-Goldwasser-Lovász-Safra-Szegedy 91, Babai-Fortnow-Levin-Szegedy 91, Arora-Safra 92, Arora-Lund-Mutwani-Sudan-Szegedy 92]



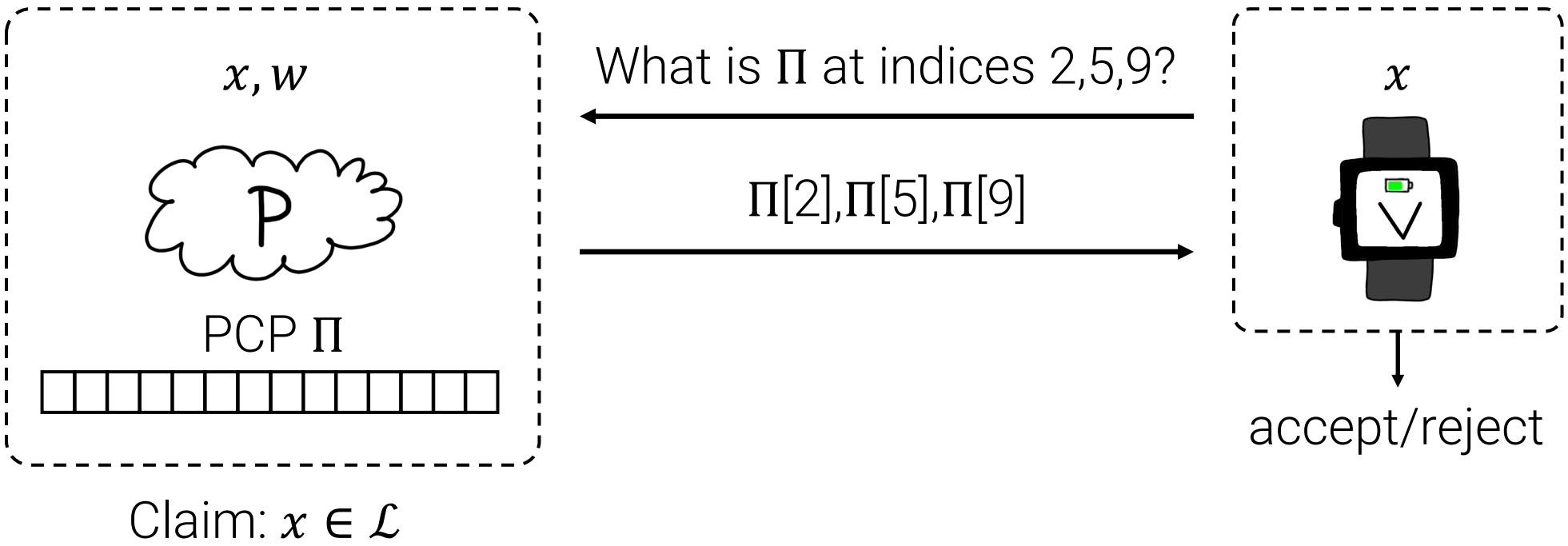
Thm. If $\mathcal{L} \in NP$, V making $polylog(n)$ queries:

- accepts if $x \in \mathcal{L}$
- rejects if $x \notin \mathcal{L}$ with $1 - negl(n)$ probability.

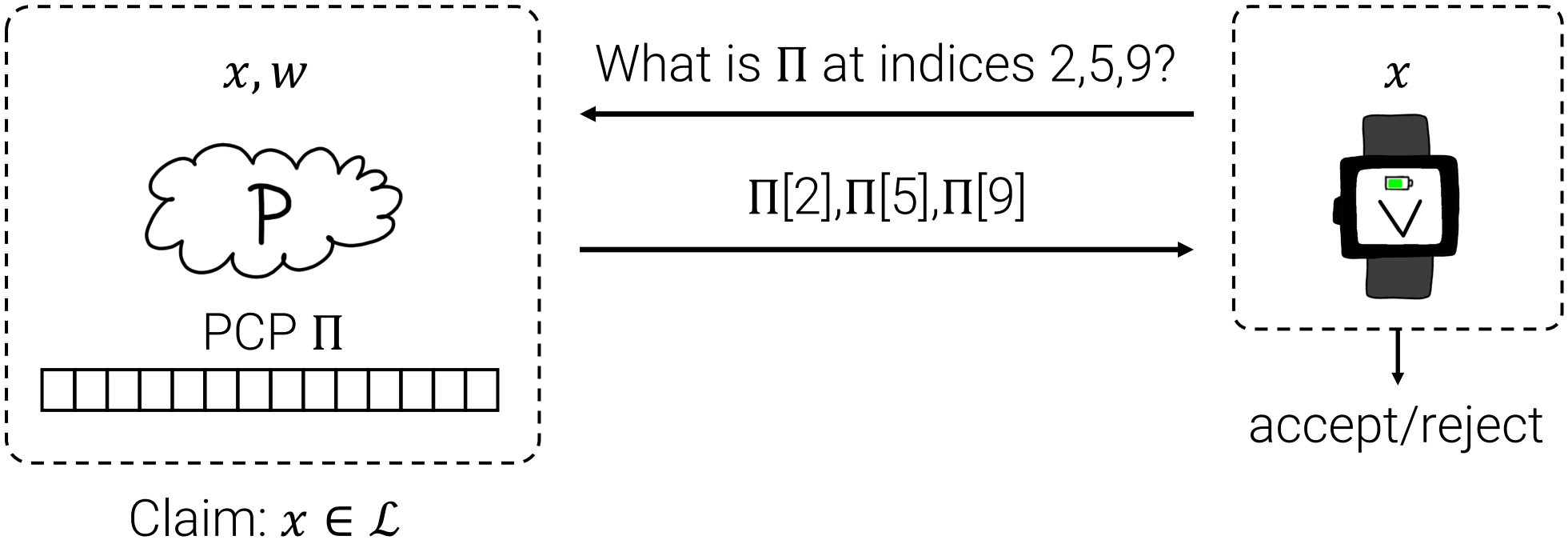
From PCPs to Arguments for NP



From PCPs to Arguments for NP

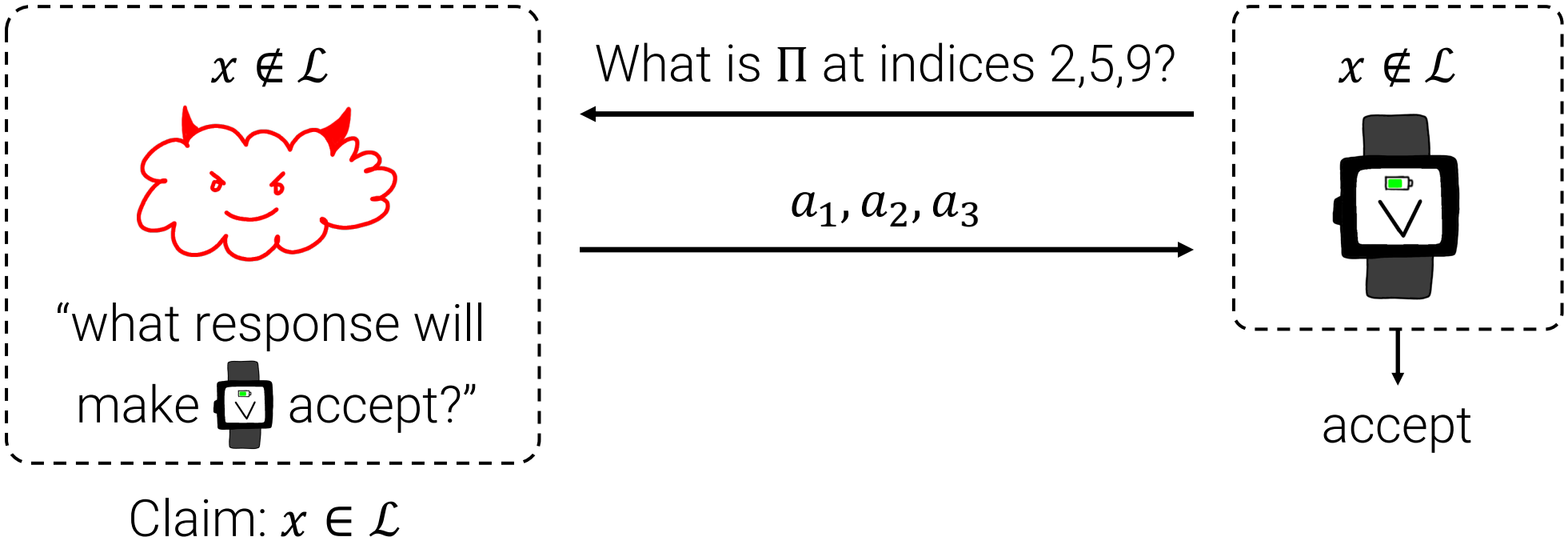


From PCPs to Arguments for NP



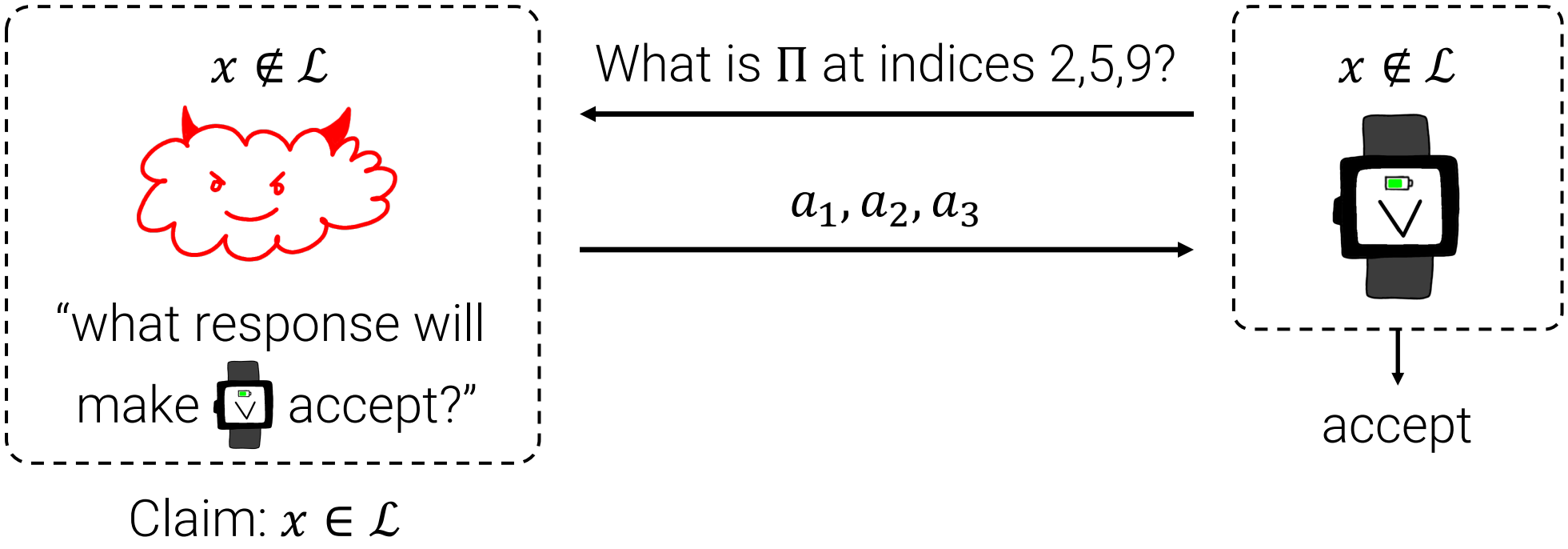
Problem:  can answer adaptively!

From PCPs to Arguments for NP



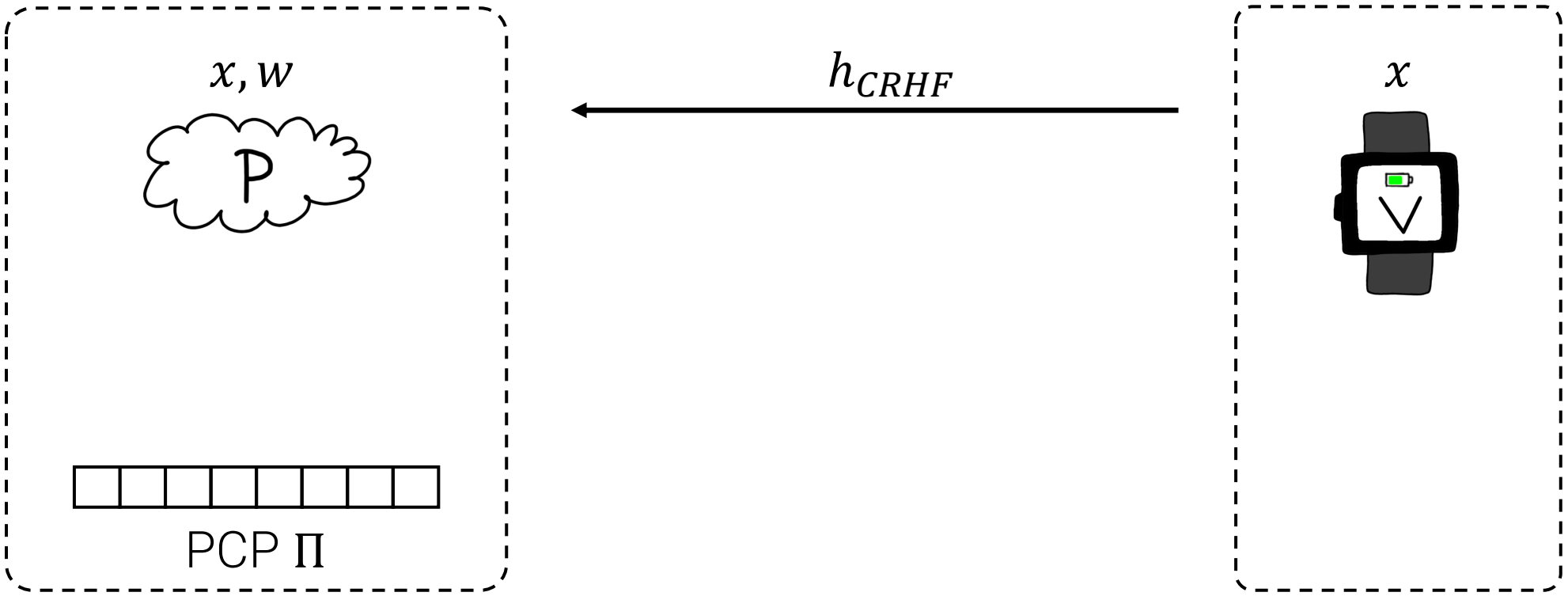
Problem: can answer adaptively!

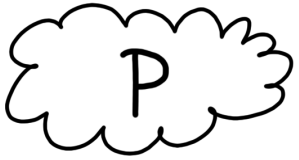
From PCPs to Arguments for NP



For soundness, must ensure doesn't change PCP after seeing message.

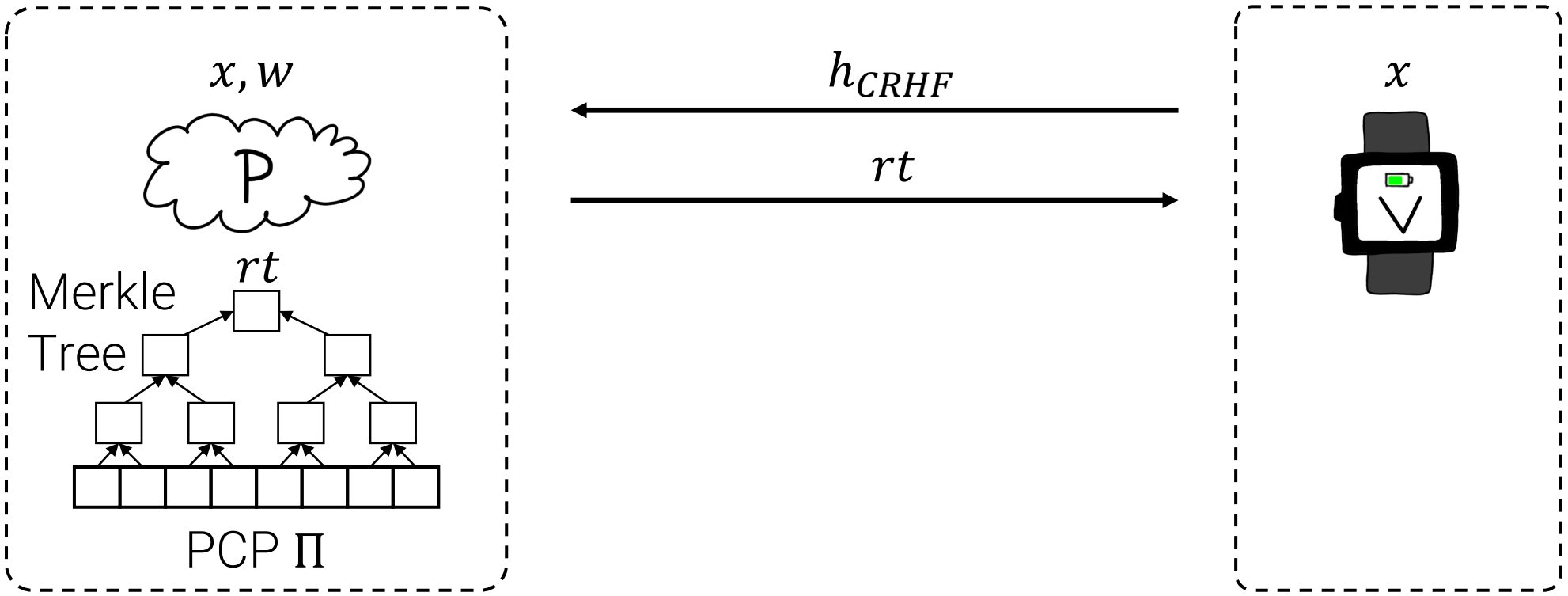
From PCPs to Delegation [Kilian 92]

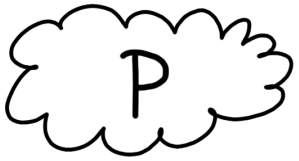


Kilian's protocol: Force 

to send short commitment to Π

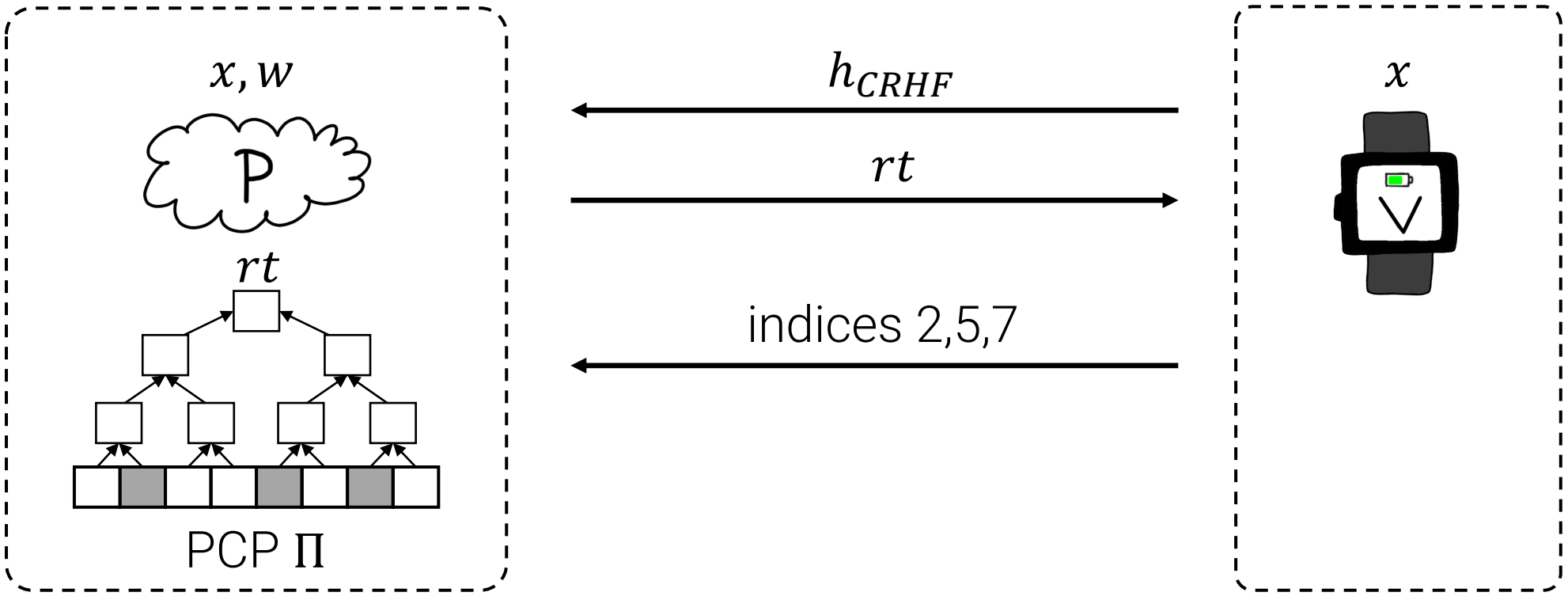
From PCPs to Delegation [Kilian 92]

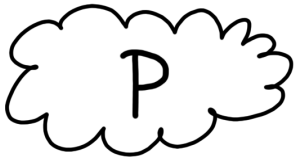


Kilian's protocol: Force 

to send short commitment to Π

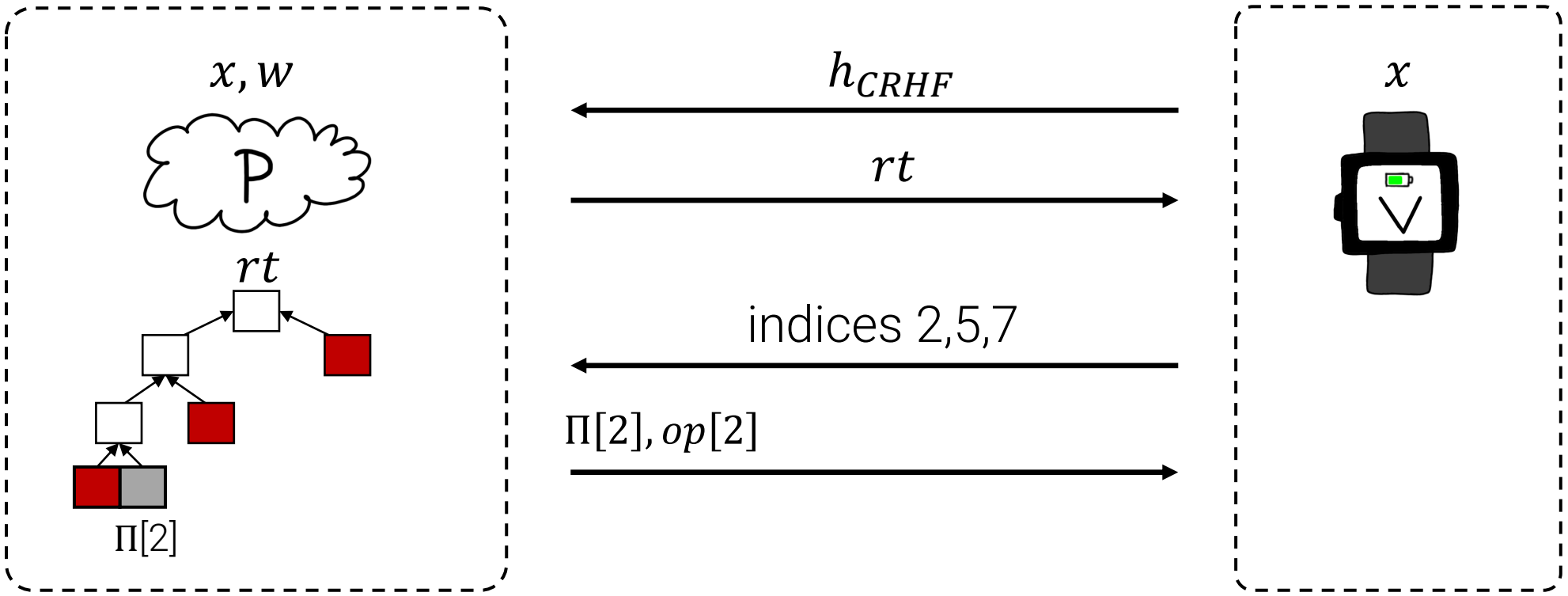
From PCPs to Delegation [Kilian 92]

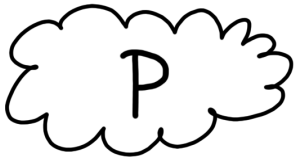


Kilian's protocol: Force 

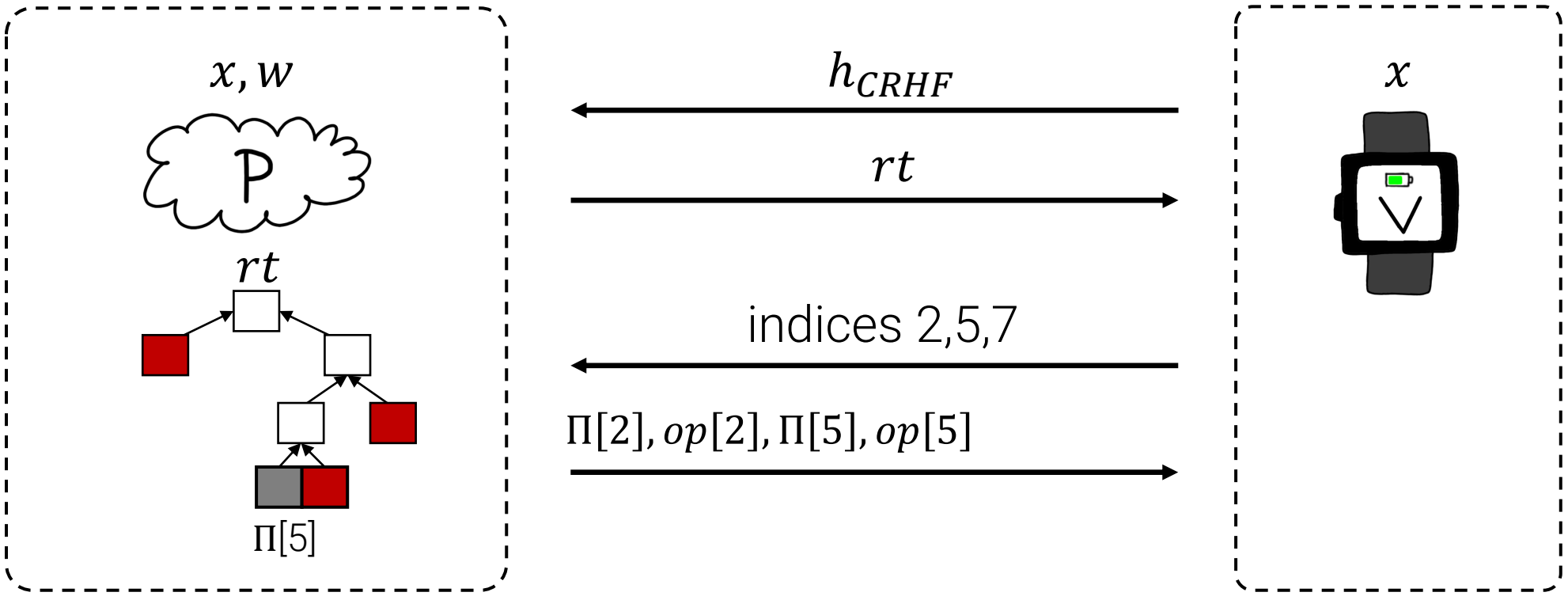
to send short commitment to Π

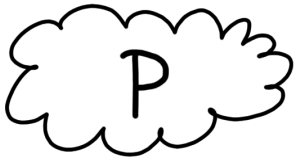
From PCPs to Delegation [Kilian 92]



Kilian's protocol: Force  to send short commitment to Π

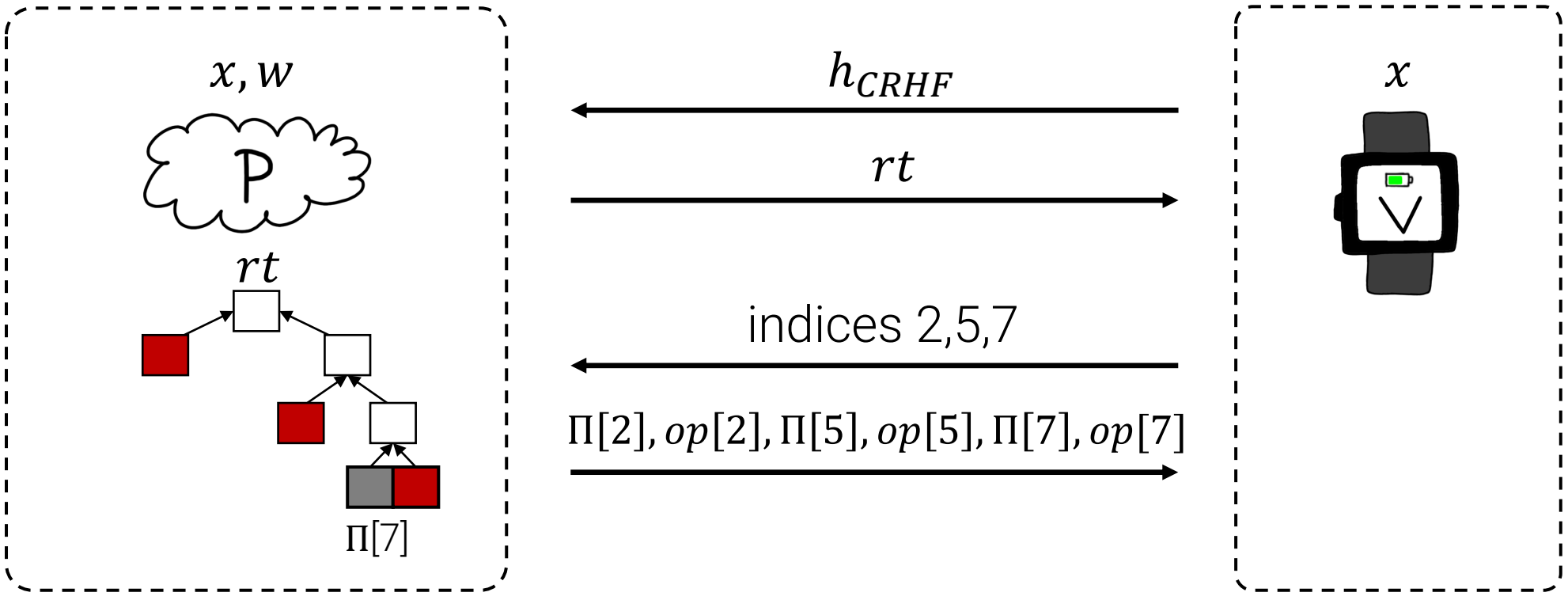
From PCPs to Delegation [Kilian 92]

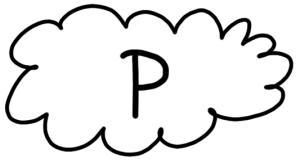


Kilian's protocol: Force 

to send short commitment to Π

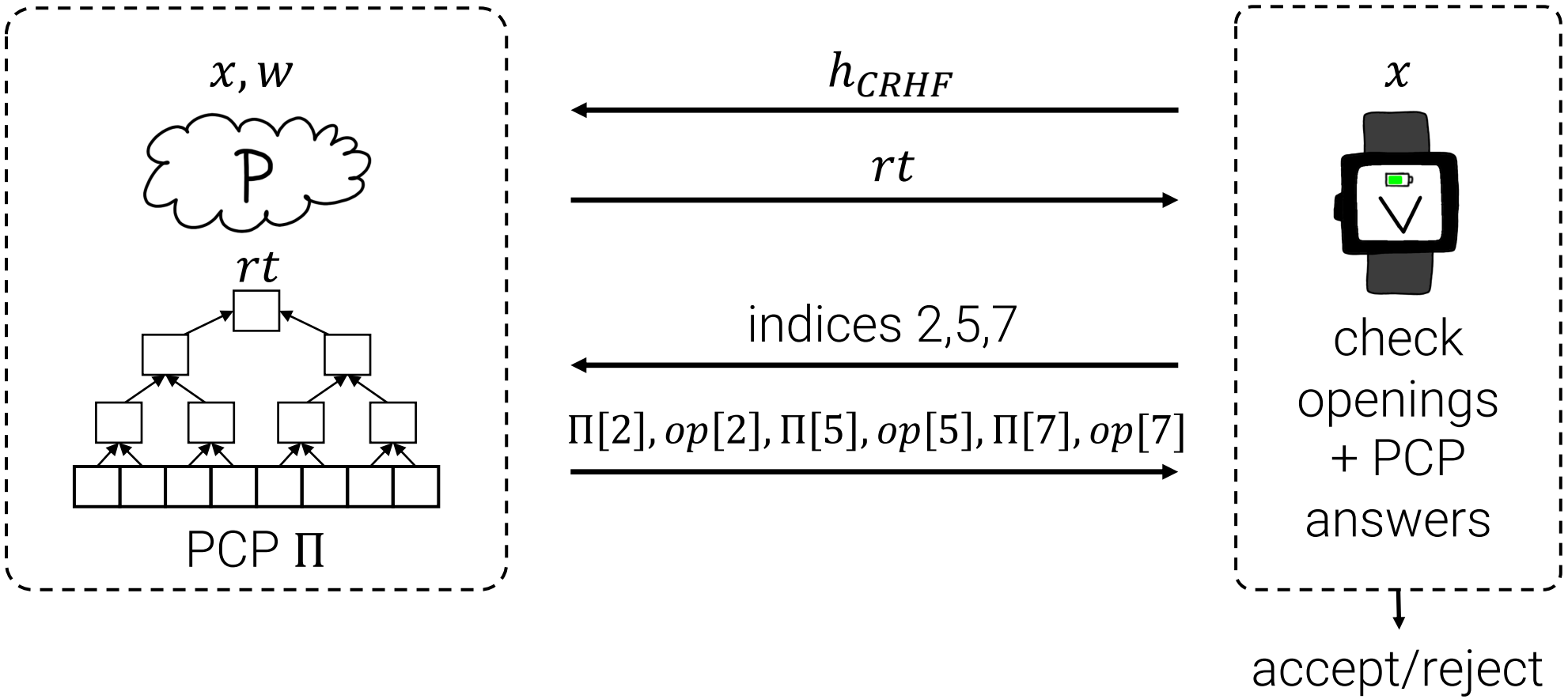
From PCPs to Delegation [Kilian 92]

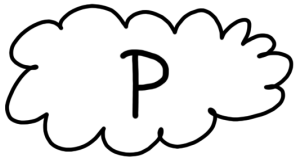


Kilian's protocol: Force 

to send short commitment to Π

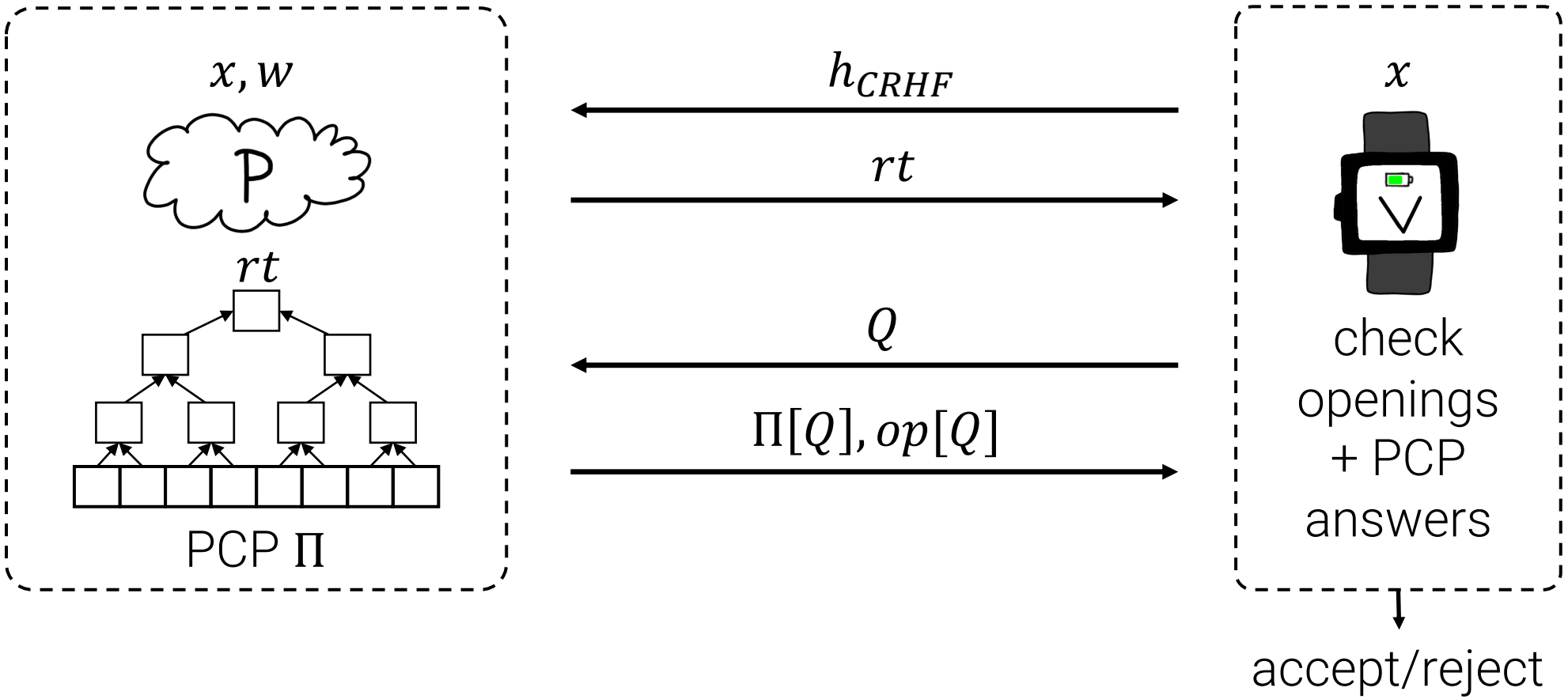
From PCPs to Delegation [Kilian 92]

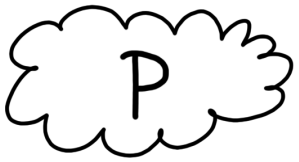


Kilian's protocol: Force 

to send short commitment to Π

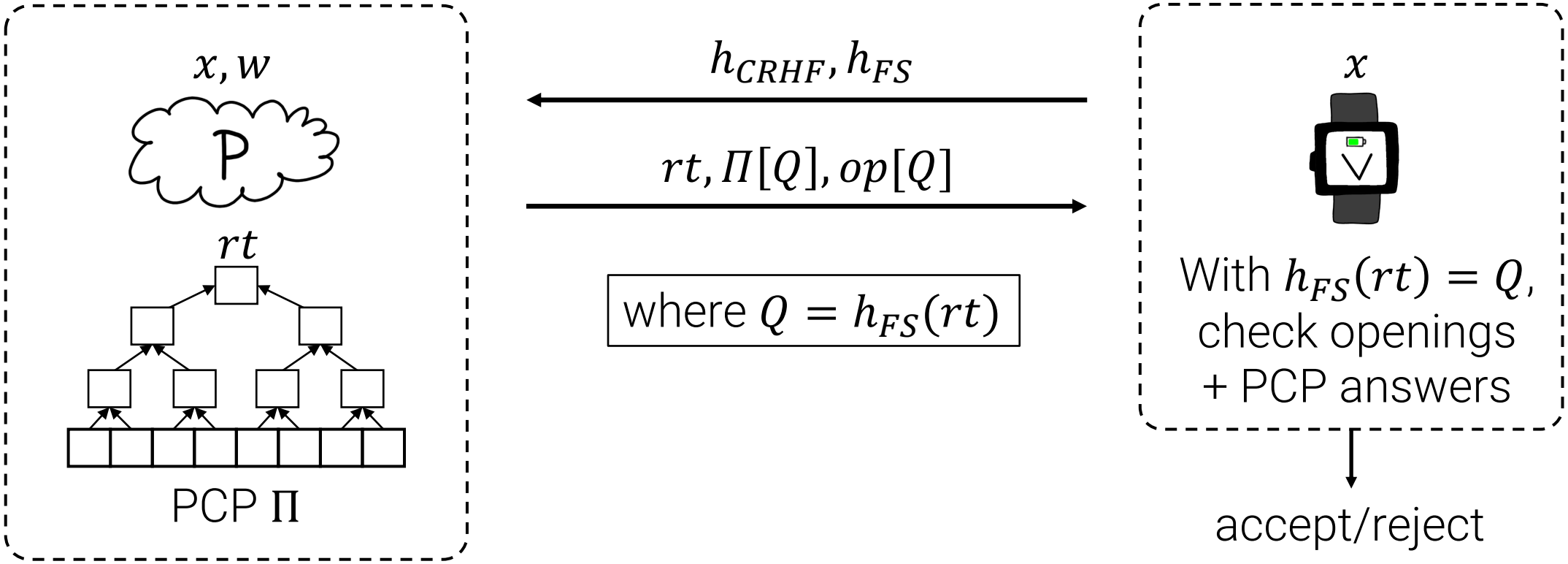
From PCPs to Delegation [Kilian 92]



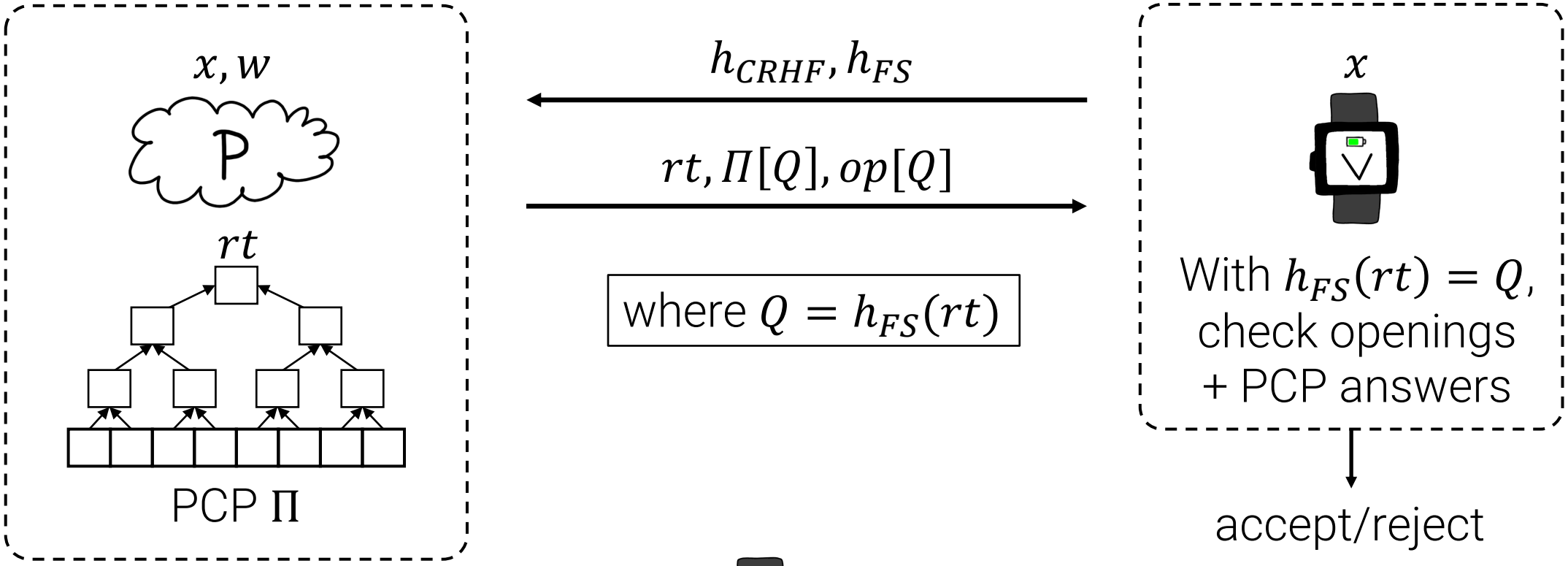
Kilian's protocol: Force 

to send short commitment to Π

Fiat Shamir + Kilian = SNARG



Fiat Shamir + Kilian = SNARG

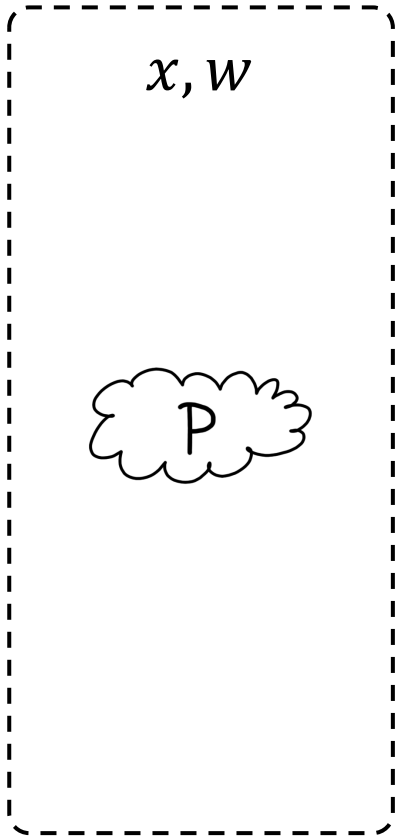


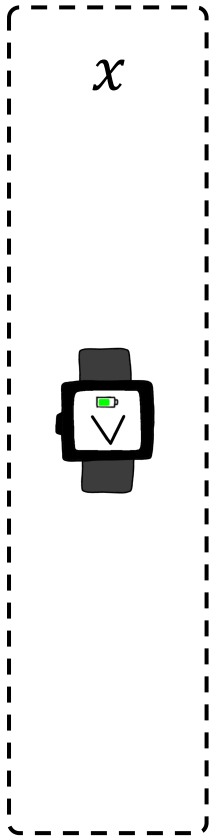
In plain model,  sends h_{CRHF}, h_{FS}

(In ROM, both are replaced by the oracle [Micali94])

Interactive Oracle Proofs (IOPs)

[Ben-Sasson-Chiesa-Spooner 16, Reingold-Rothblum-Rothblum 16]

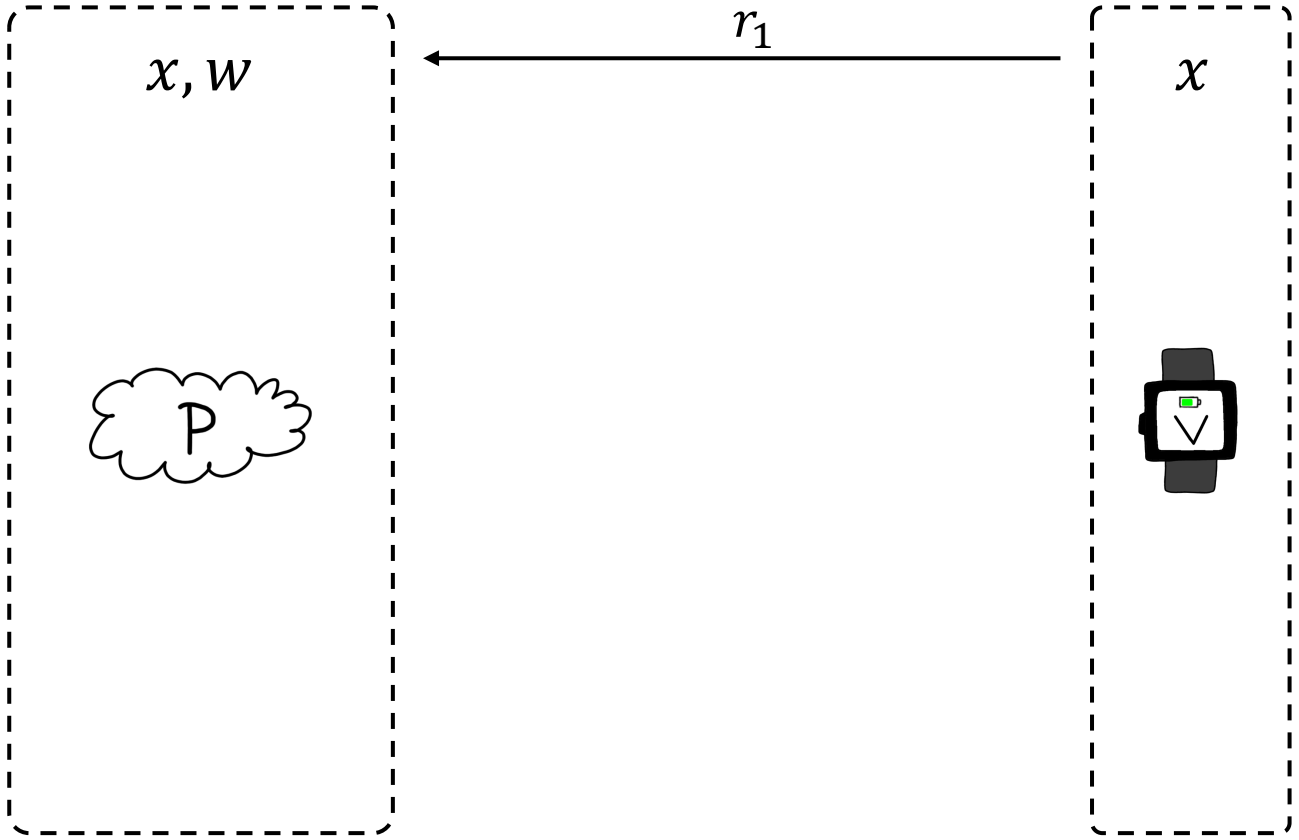




Claim: $x \in \mathcal{L}$

Interactive Oracle Proofs (IOPs)

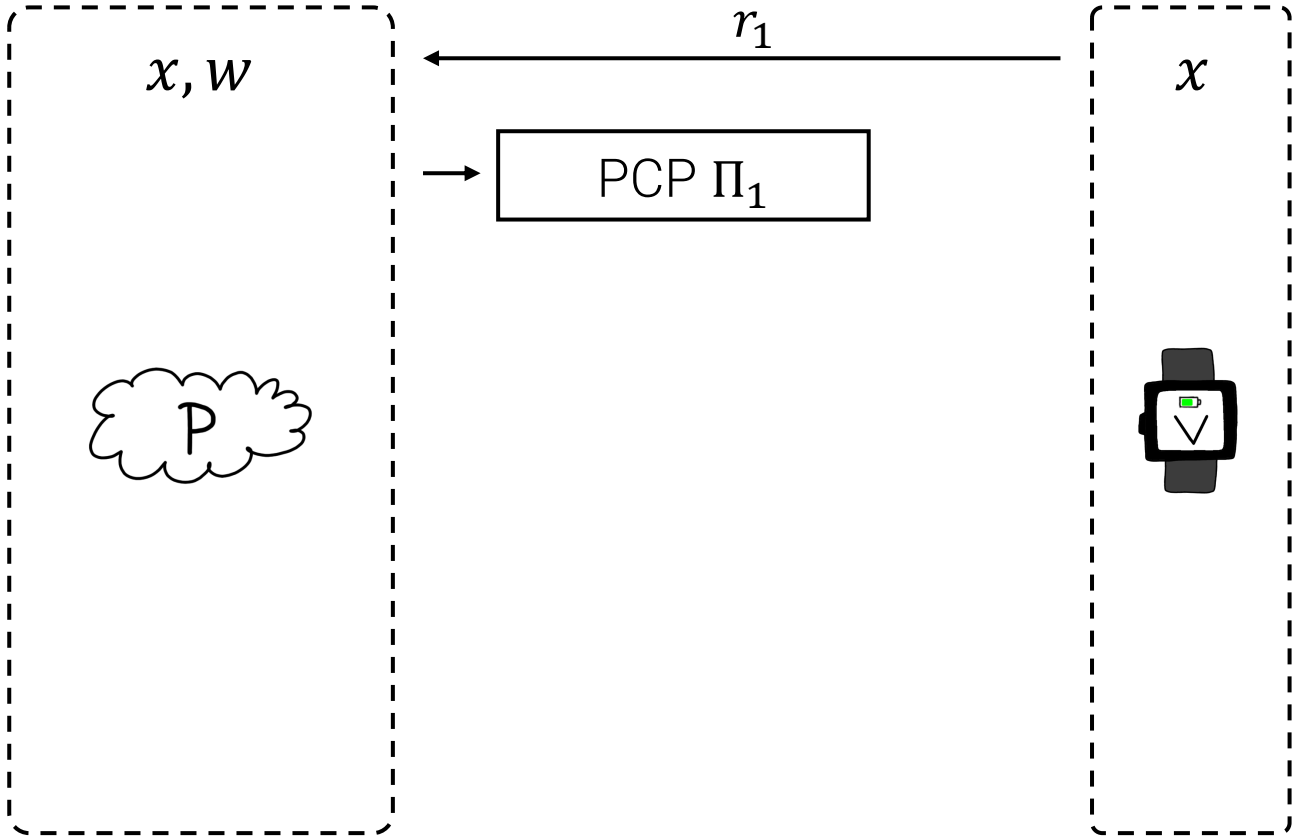
[Ben-Sasson-Chiesa-Spooner 16, Reingold-Rothblum-Rothblum 16]



Claim: $x \in \mathcal{L}$

Interactive Oracle Proofs (IOPs)

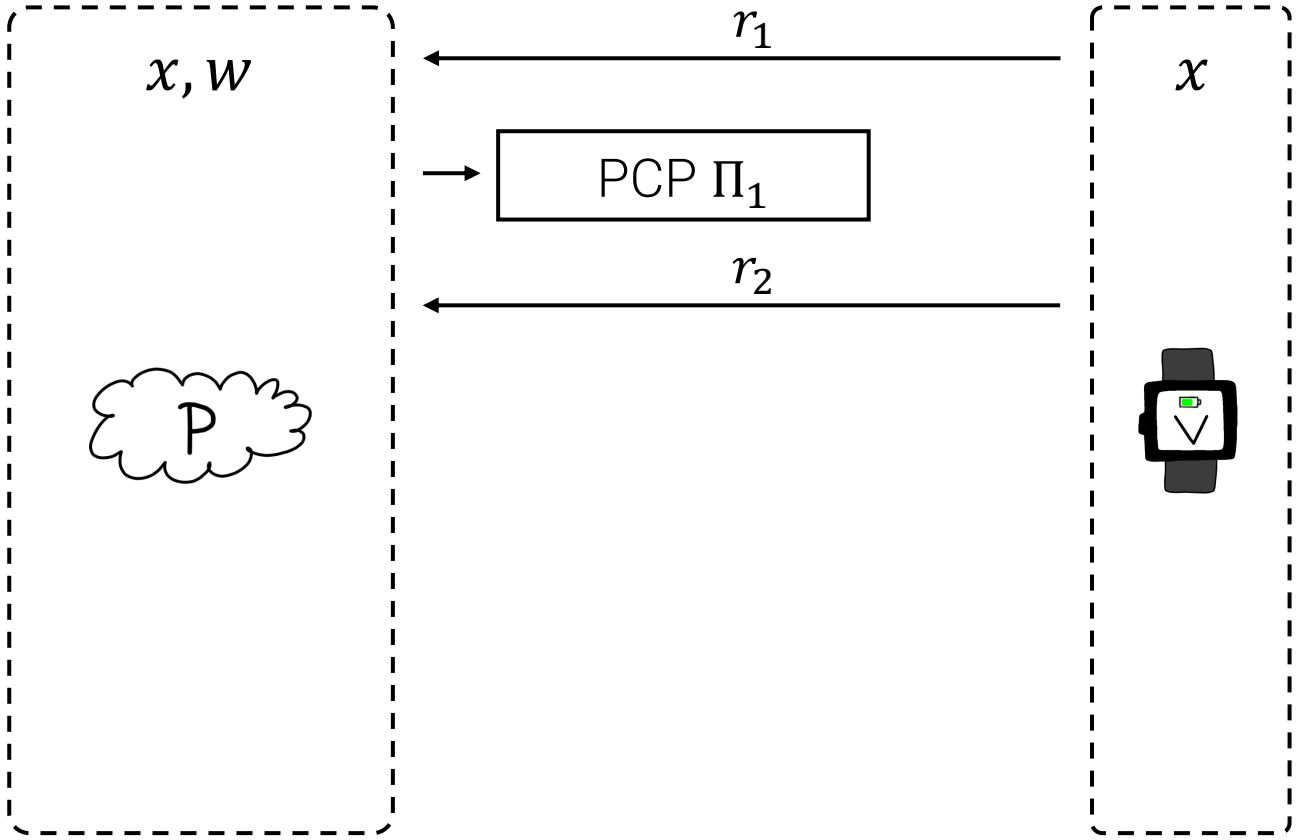
[Ben-Sasson-Chiesa-Spooner 16, Reingold-Rothblum-Rothblum 16]



Claim: $x \in \mathcal{L}$

Interactive Oracle Proofs (IOPs)

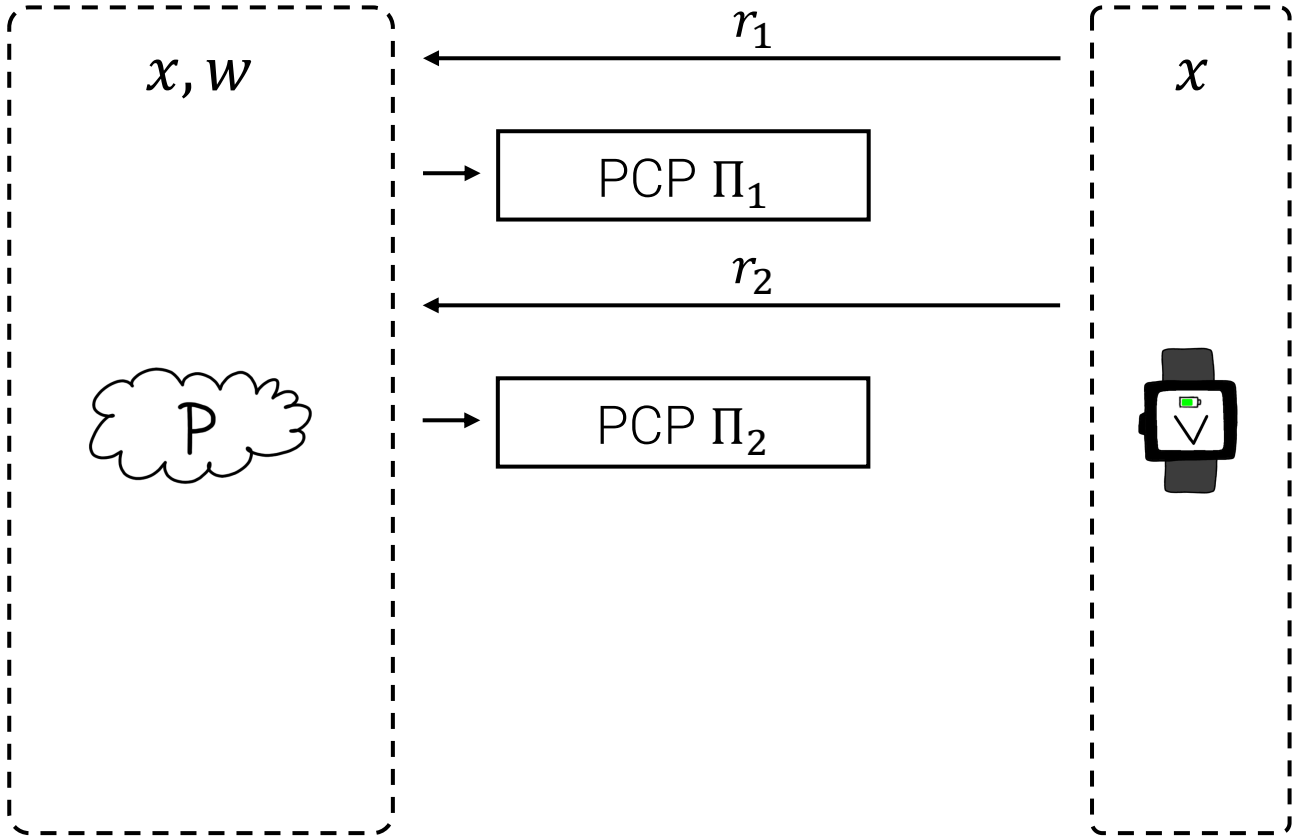
[Ben-Sasson-Chiesa-Spooner 16, Reingold-Rothblum-Rothblum 16]



Claim: $x \in \mathcal{L}$

Interactive Oracle Proofs (IOPs)

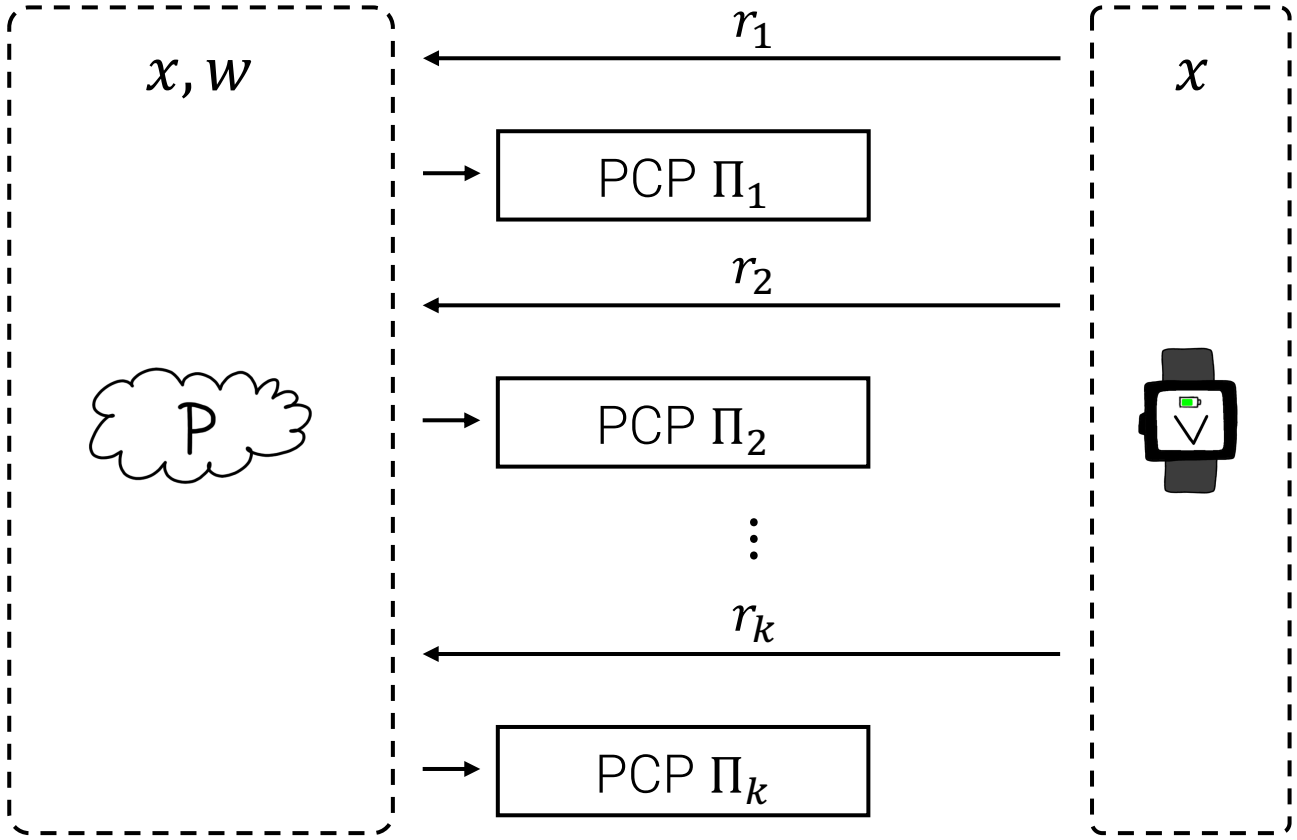
[Ben-Sasson-Chiesa-Spooner 16, Reingold-Rothblum-Rothblum 16]



Claim: $x \in \mathcal{L}$

Interactive Oracle Proofs (IOPs)

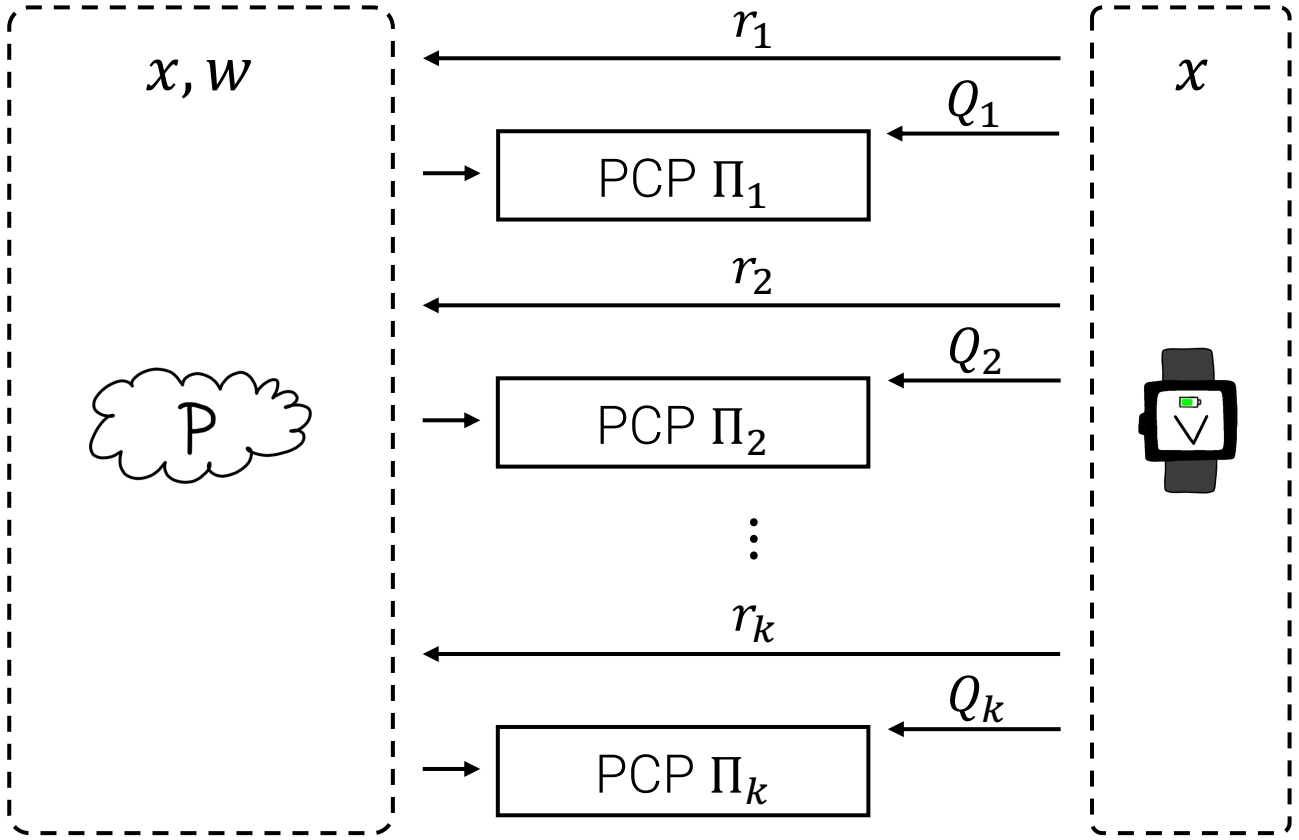
[Ben-Sasson-Chiesa-Spooner 16, Reingold-Rothblum-Rothblum 16]



Claim: $x \in \mathcal{L}$

Interactive Oracle Proofs (IOPs)

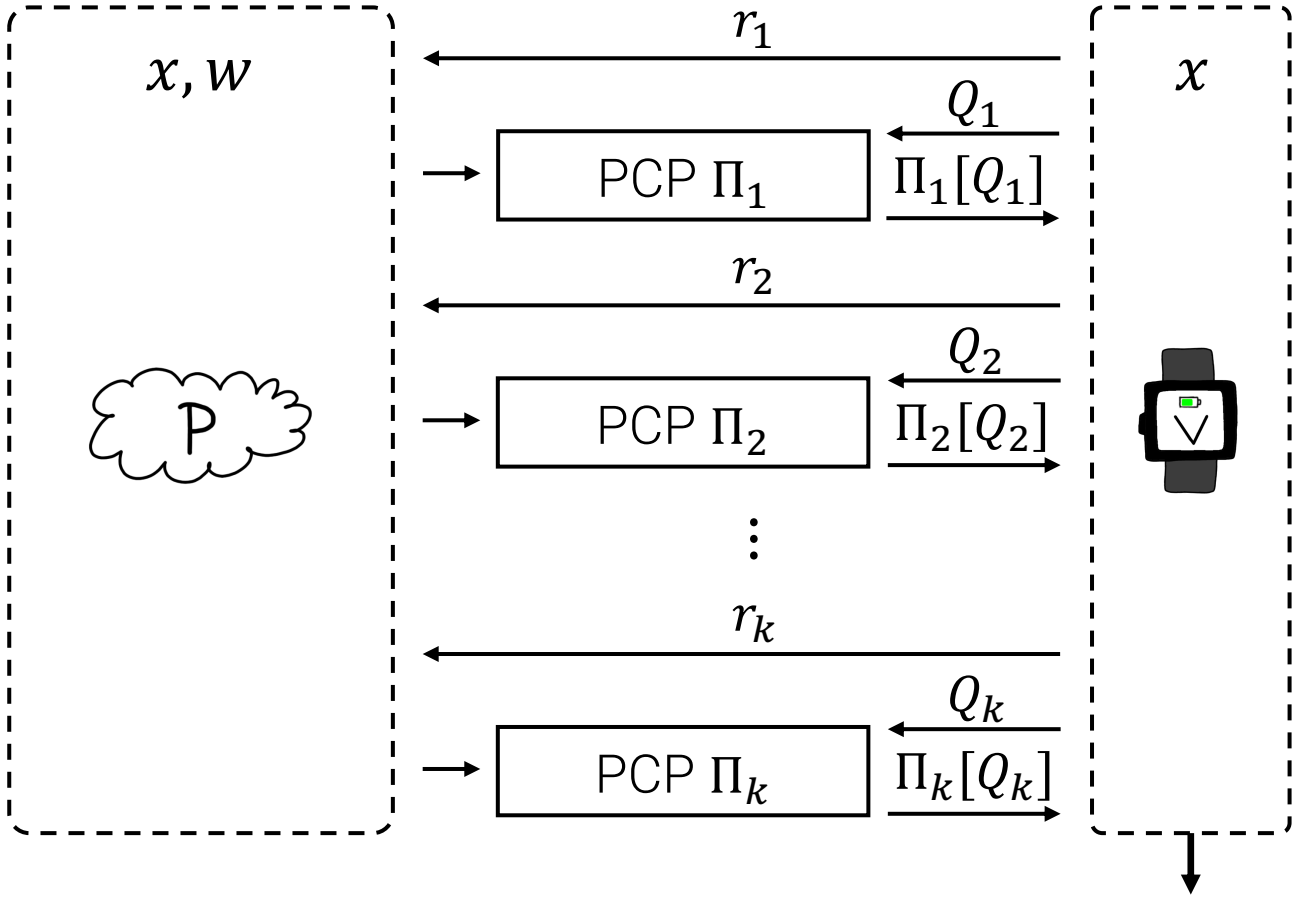
[Ben-Sasson-Chiesa-Spooner 16, Reingold-Rothblum-Rothblum 16]



Claim: $x \in \mathcal{L}$

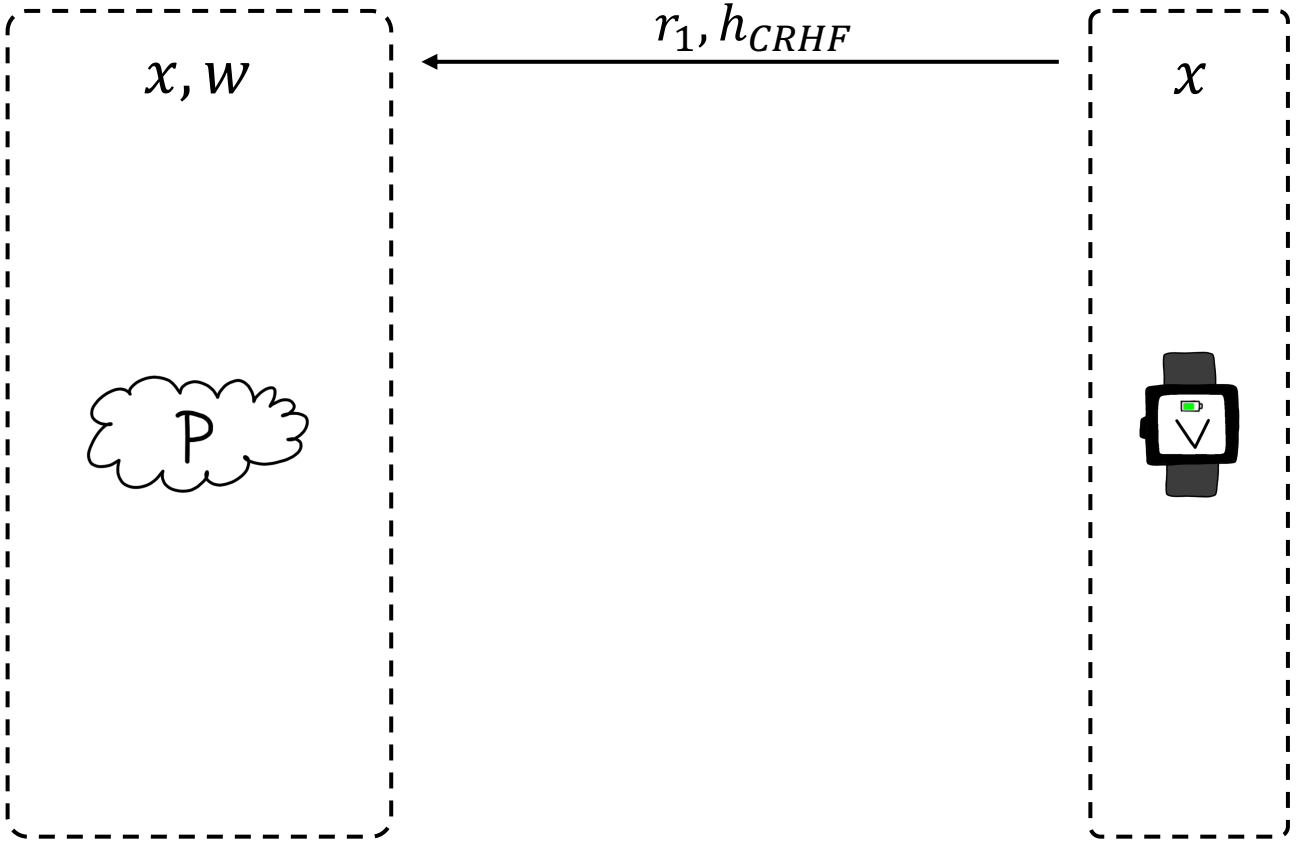
Interactive Oracle Proofs (IOPs)

[Ben-Sasson-Chiesa-Spooner 16, Reingold-Rothblum-Rothblum 16]



Kilian for IOPs

[Ben-Sasson-Chiesa-Spooner 16]



Claim: $x \in \mathcal{L}$

Same idea as Kilian for PCPs:

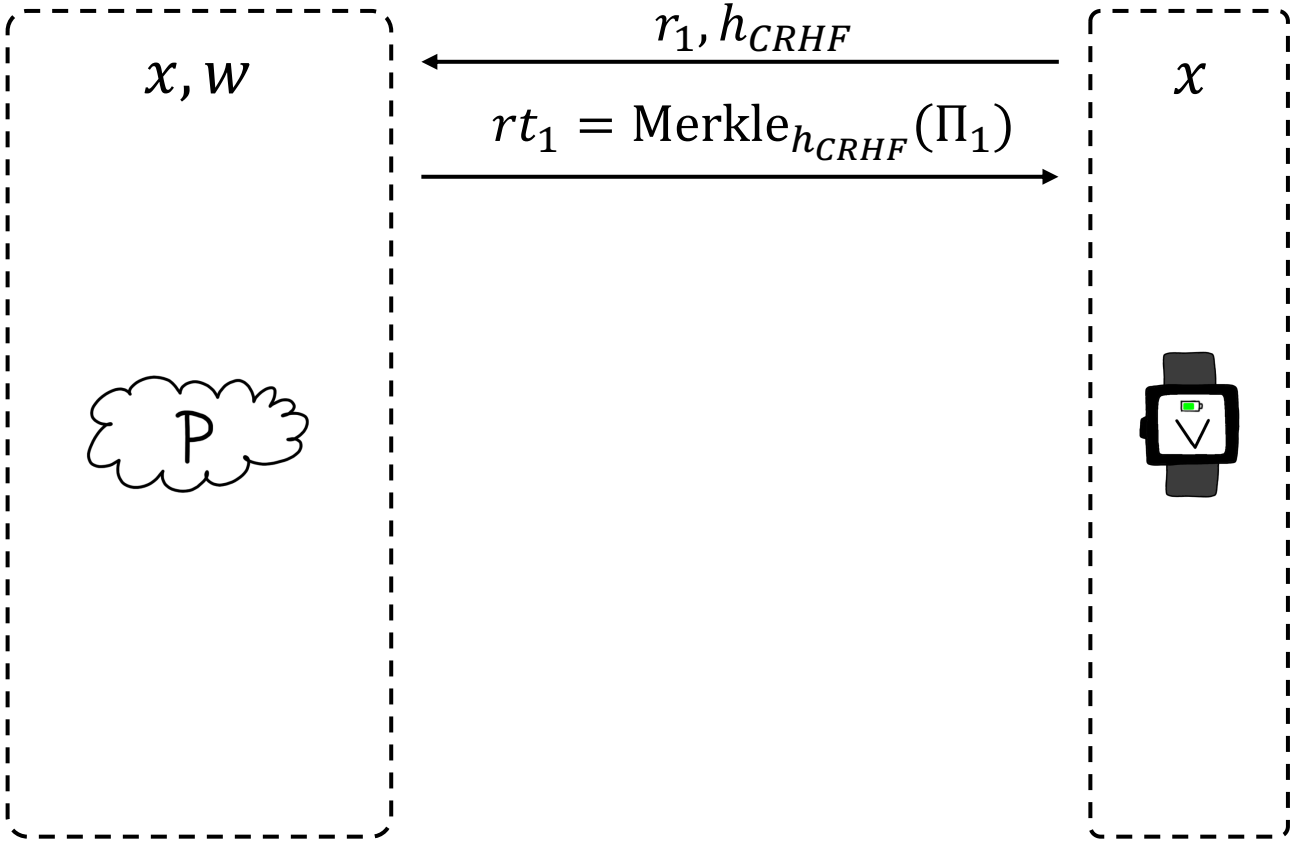
-  sends h_{CRHF}
-  commits to Π_i with

$$rt_i = \mathbf{Merkle}_{h_{CRHF}}(\Pi_i)$$

-  accepts if IOP verifier accepts and openings valid

Kilian for IOPs

[Ben-Sasson-Chiesa-Spooner 16]



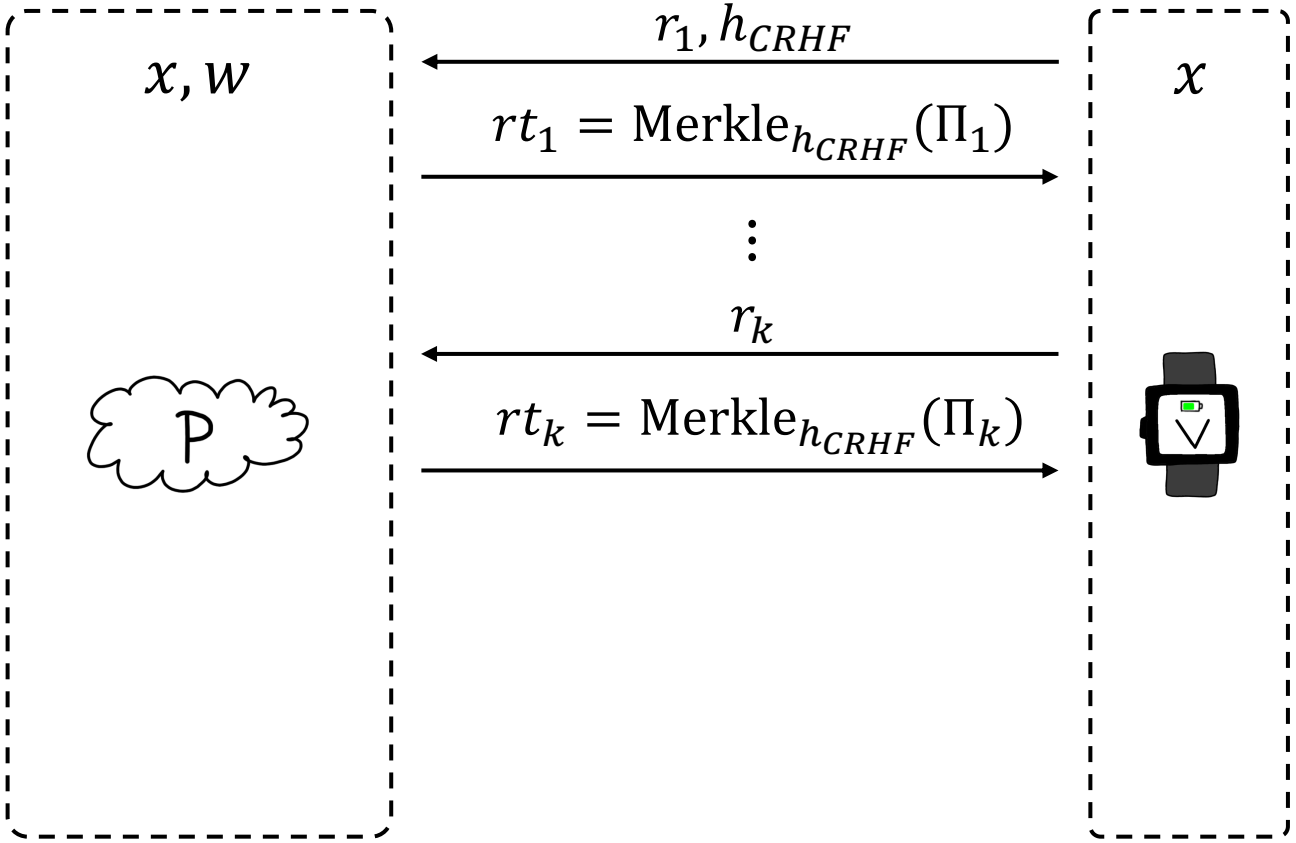
Claim: $x \in \mathcal{L}$

Same idea as Kilian for PCPs:

- V sends h_{CRHF}
- P commits to Π_i with $rt_i = \text{Merkle}_{h_{CRHF}}(\Pi_i)$
- V accepts if IOP verifier accepts and openings valid

Kilian for IOPs

[Ben-Sasson-Chiesa-Spooner 16]



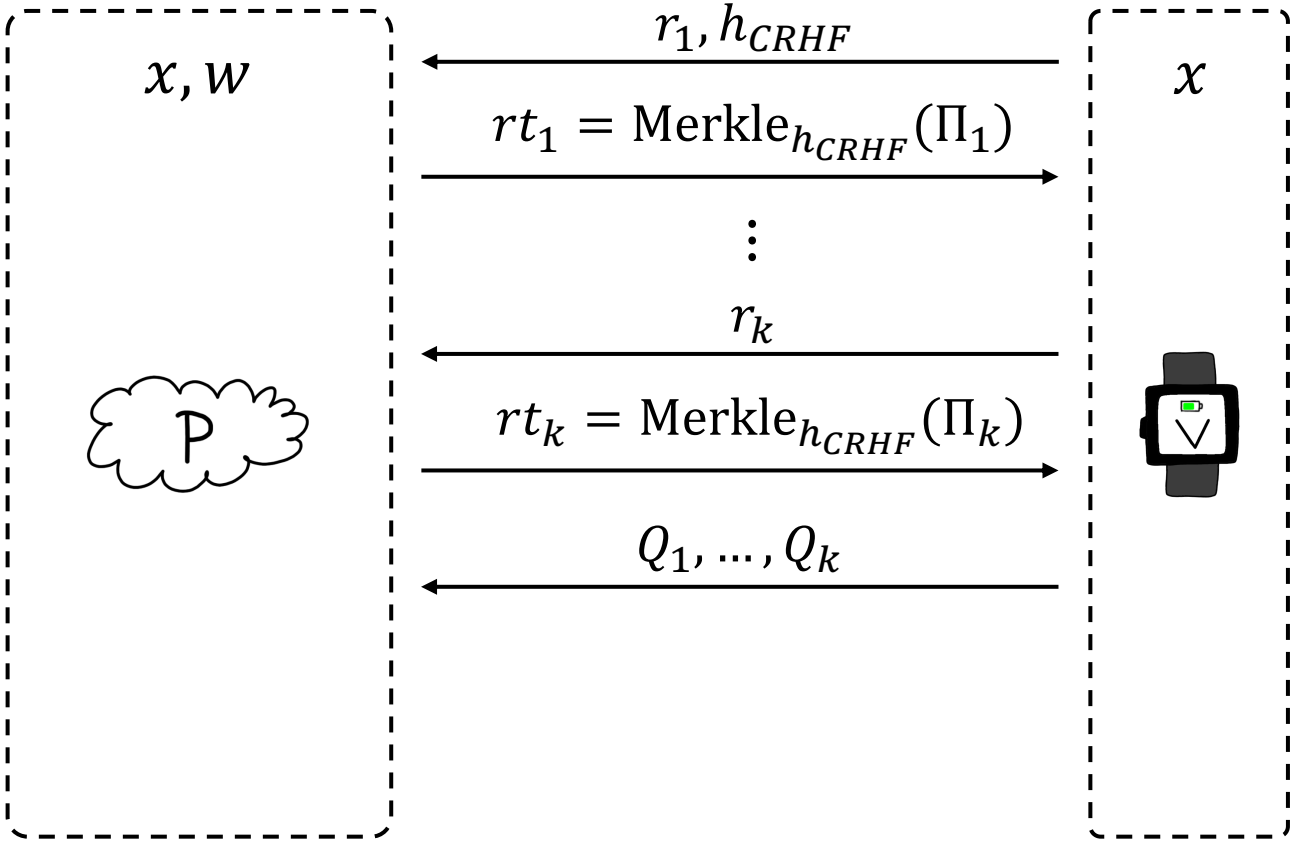
Claim: $x \in \mathcal{L}$

Same idea as Kilian for PCPs:

- V sends h_{CRHF}
- P commits to Π_i with $rt_i = \text{Merkle}_{h_{CRHF}}(\Pi_i)$
- V accepts if IOP verifier accepts and openings valid

Kilian for IOPs

[Ben-Sasson-Chiesa-Spooner 16]



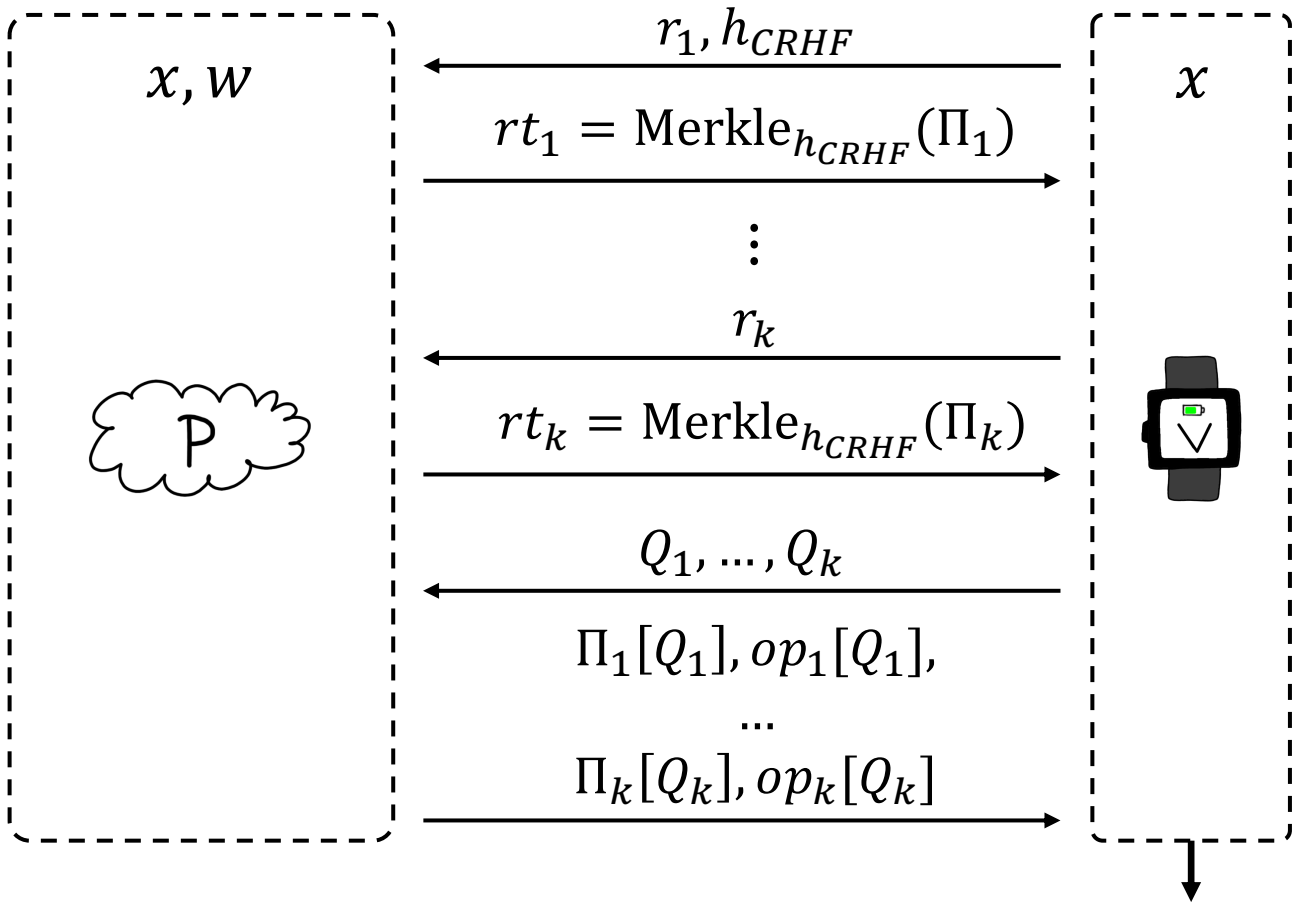
Claim: $x \in \mathcal{L}$

Same idea as Kilian for PCPs:

- sends h_{CRHF}
- P commits to Π_i with $rt_i = \text{Merkle}_{h_{CRHF}}(\Pi_i)$
- accepts if IOP verifier accepts and openings valid

Kilian for IOPs

[Ben-Sasson-Chiesa-Spooner 16]



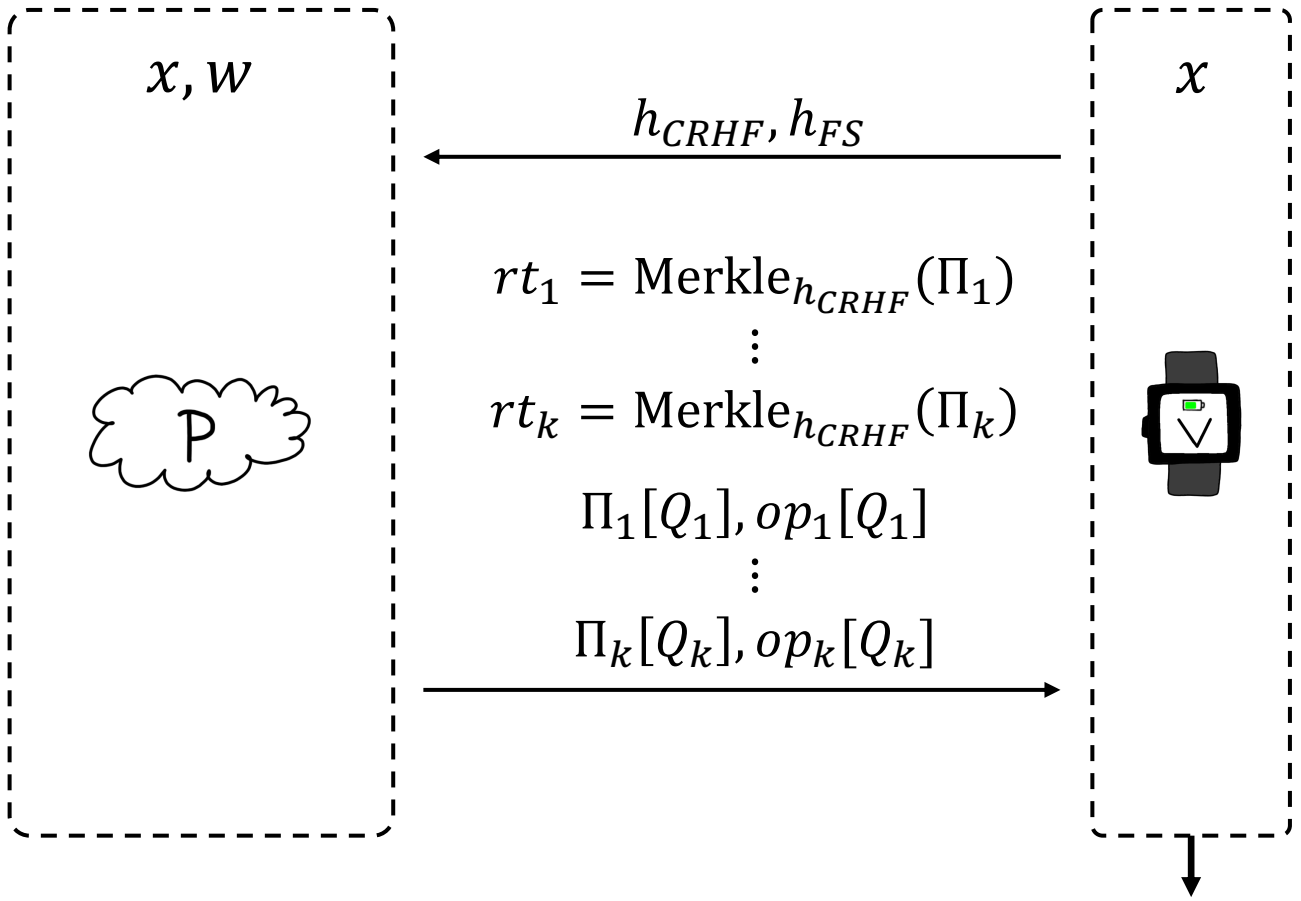
Same idea as Kilian for PCPs:

-  sends h_{CRHF}
-  commits to Π_i with

  $rt_i = \text{Merkle}_{h_{CRHF}}(\Pi_i)$
-  accepts if IOP verifier accepts and openings valid

IOP-based SNARGs

[Ben-Sasson-Chiesa-Spooner 16]

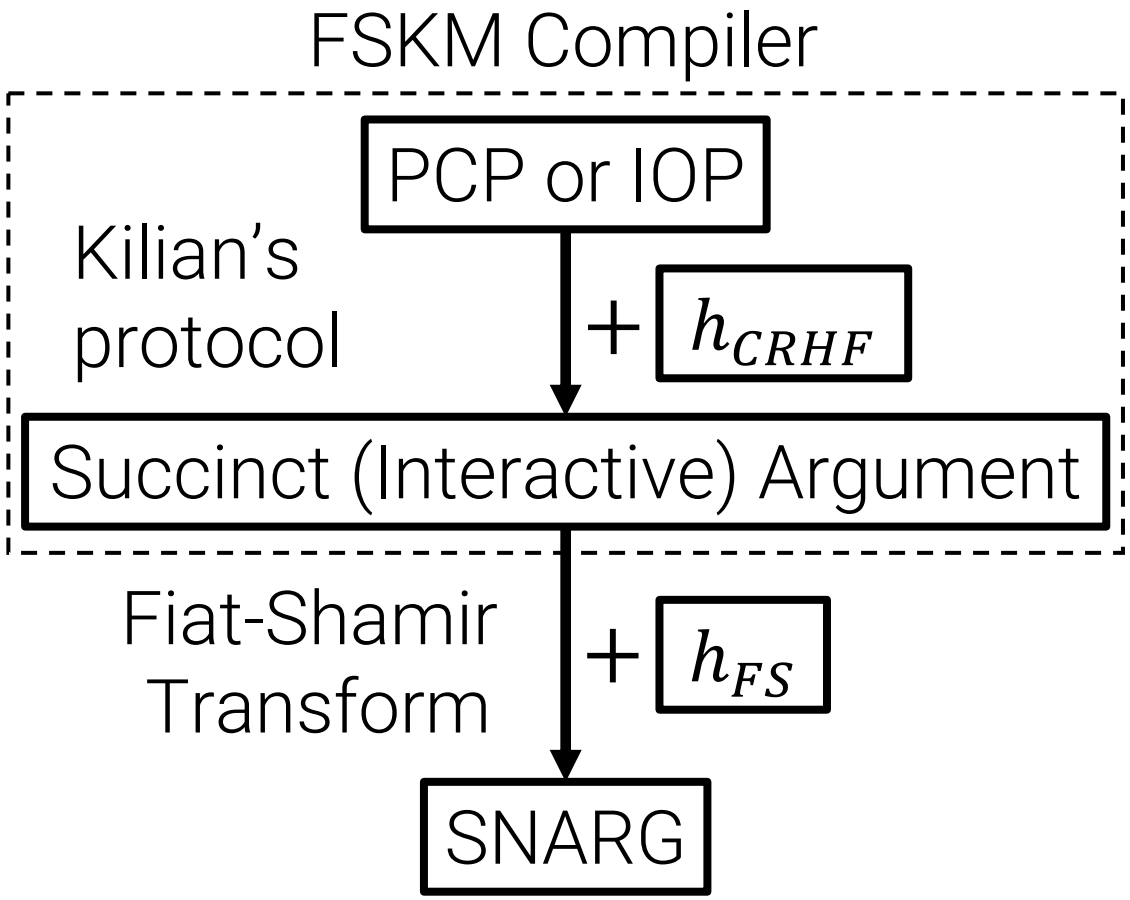


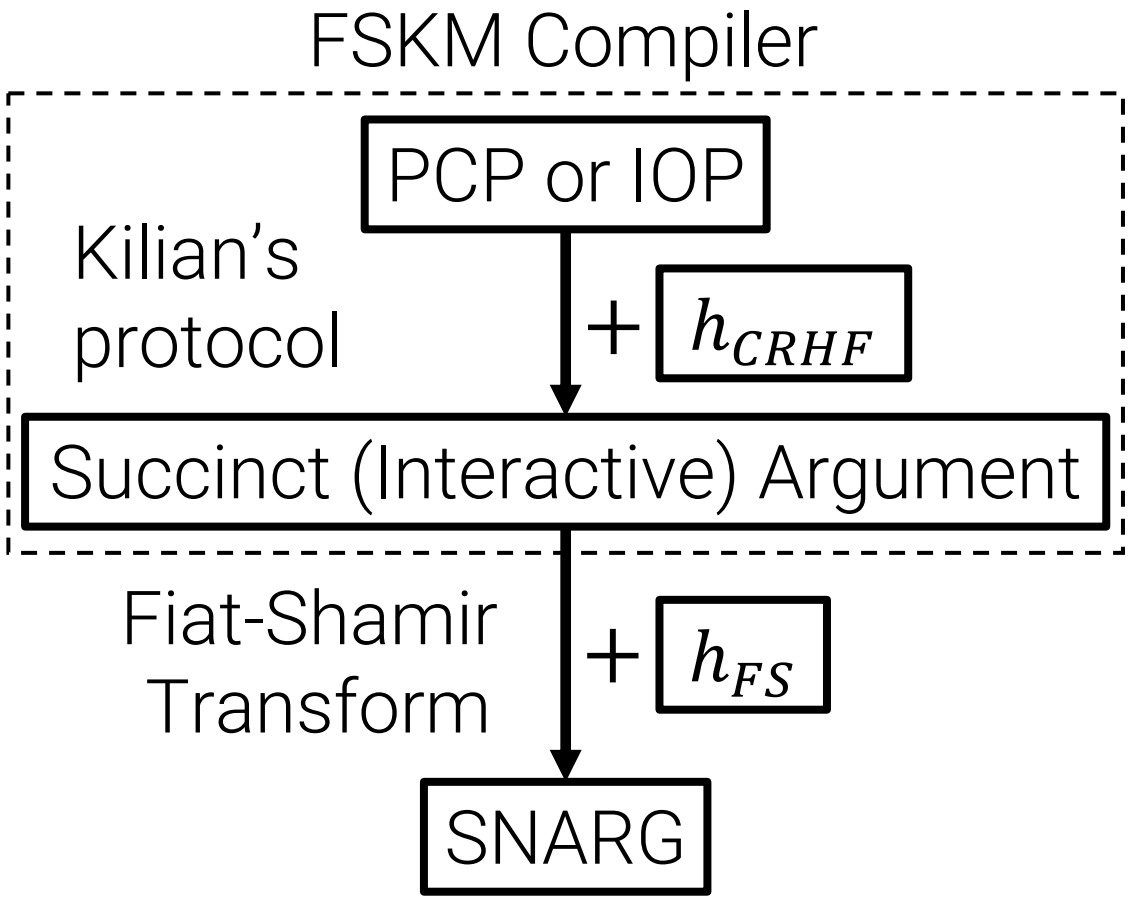
Apply Fiat-Shamir:

P generates 

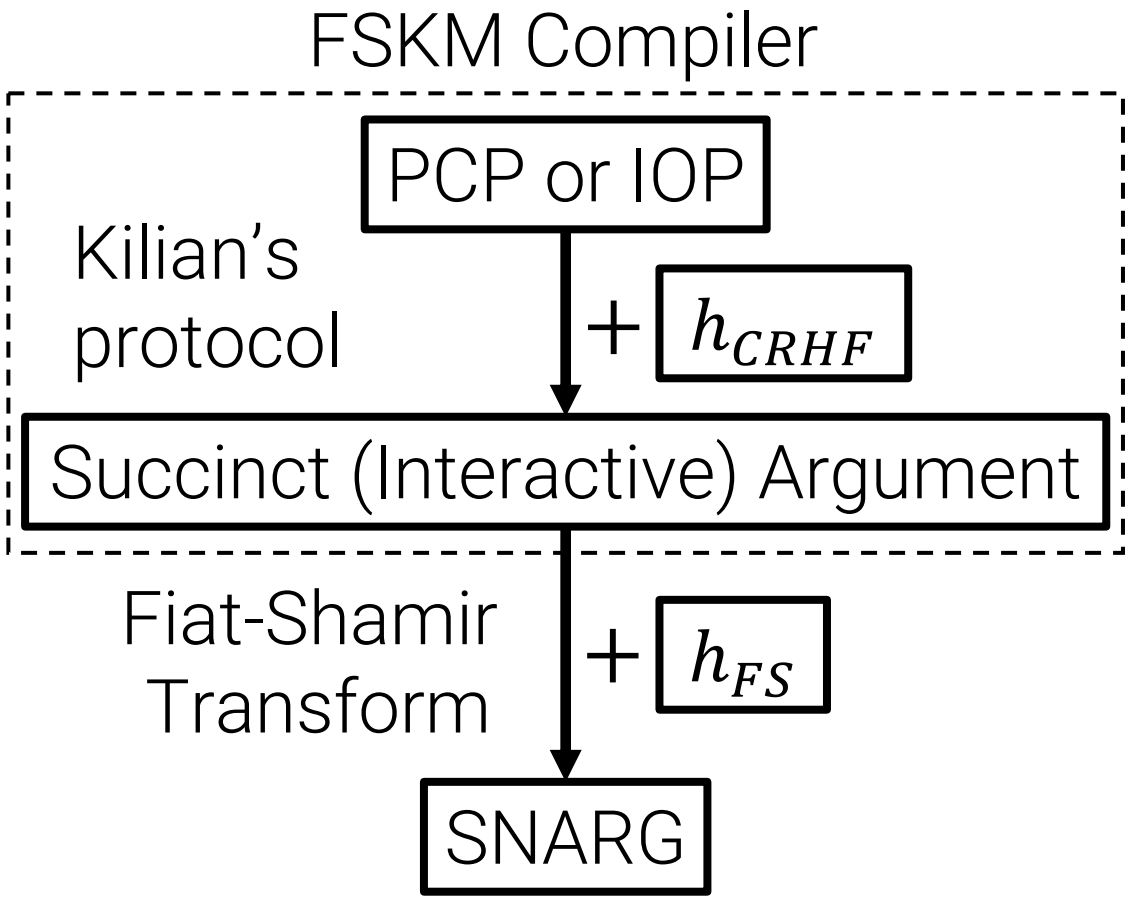
messages $r_1, \ldots, r_k, Q_1, \ldots, Q_k$

by applying h_{FS} to transcript.


FSKM Compiler
PCP or IOP
Kilian's protocol
+
h_{CRHF}
Succinct (Interactive) Argument
Fiat-Shamir Transform
+
h_{FS}
SNARG



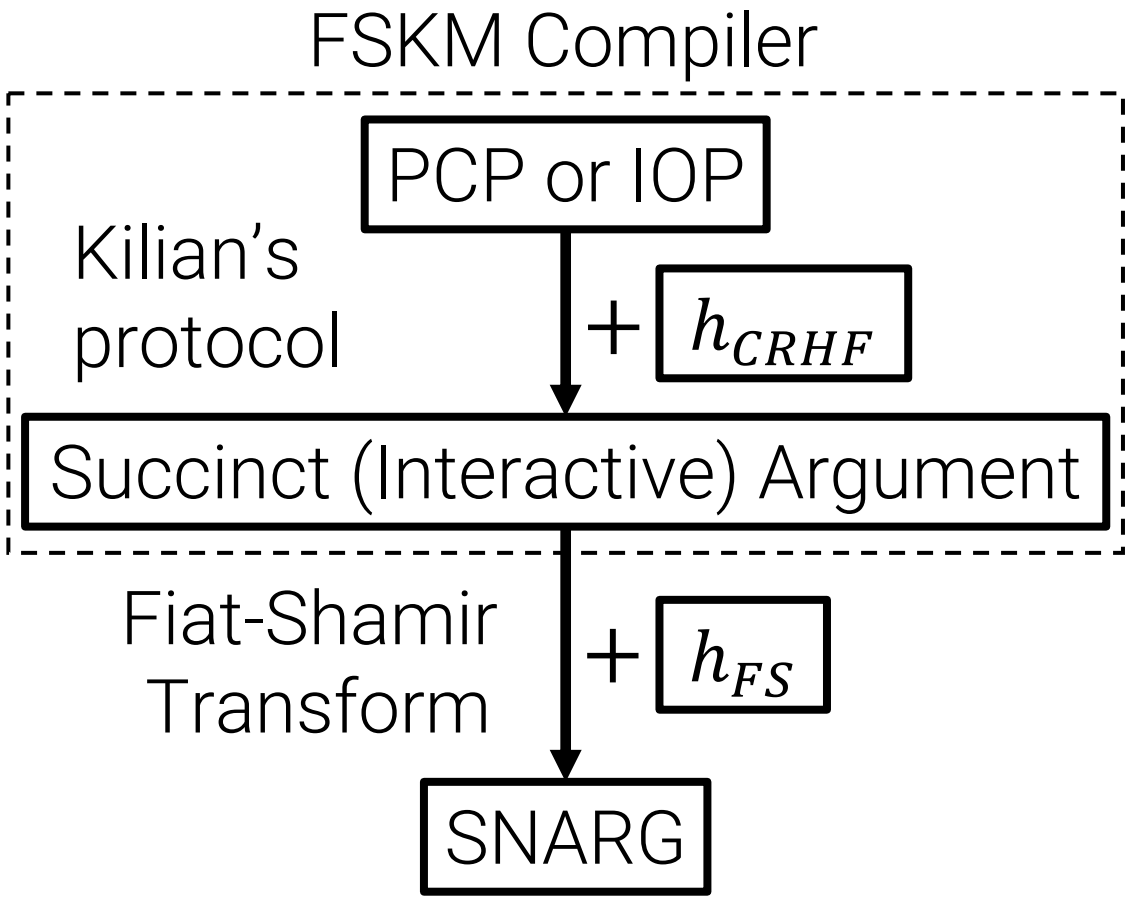
[Micali 94, Ben-Sasson-Chiesa-Spooner 16]
Sound in ROM for PCPs and IOPs with state restoration soundness



[Micali 94, Ben-Sasson-Chiesa-Spooner 16]
Sound in ROM for PCPs and IOPs with state restoration soundness

[Result 1] There exists "bad" two-round IOP s.t. FSKM is unsound for any h_{CRHF} and any h_{FS}.

Takeaway: FSKM cannot generically turn IOPs into SNARGs (in the plain model).



[Micali 94, Ben-Sasson-Chiesa-Spooner 16]
Sound in ROM for PCPs and IOPs with state restoration soundness

[Result 1] There exists "bad" two-round IOP s.t. FSKM is unsound for any h_{CRHF} and any h_{FS}.

Takeaway: FSKM cannot generically turn IOPs into SNARGs (in the plain model).

[Result 2] There exists* "bad" h_{CRHF} s.t. FSKM is unsound for (almost) any PCP and any h_{FS}.

Takeaway: No "good enough" h_{FS} for Kilian -> SNARG

*need "computationally-unique" SNARKs

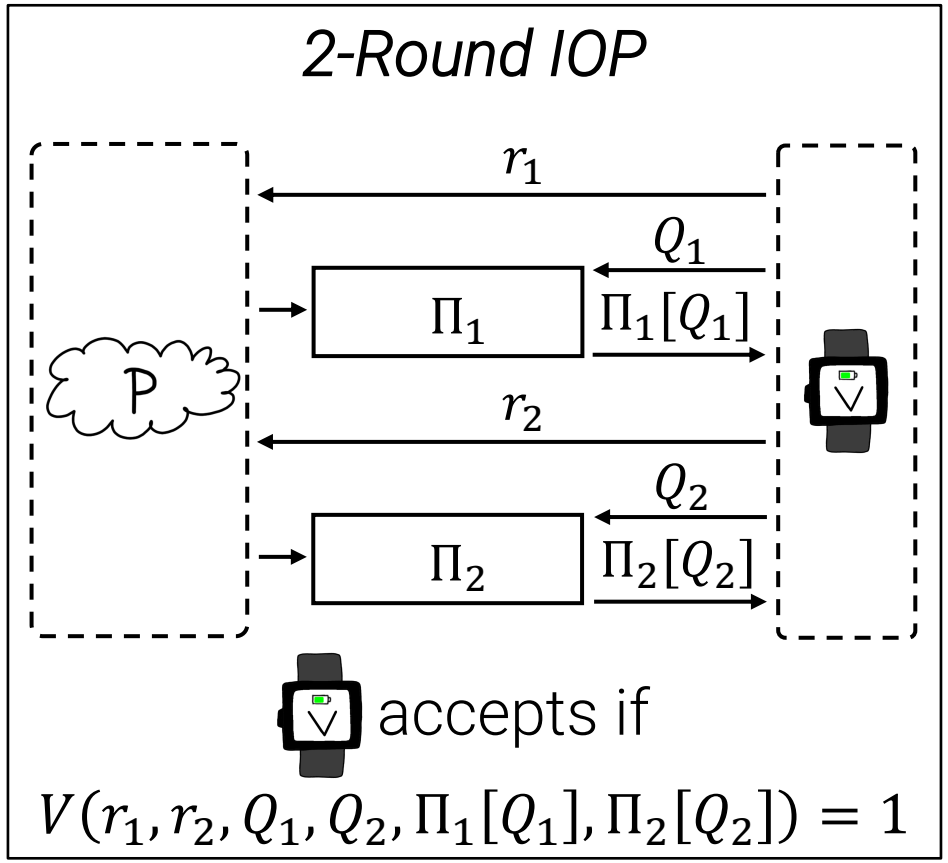
Outline

- (Review) Kilian's Protocol
- A Bad IOP for IOP-Based SNARGs
- (If time) A Bad CRHF for PCP-Based SNARGs

Result: For any $\mathcal{L} \in NP$, there is an FSKM-incompatible 2-round IOP

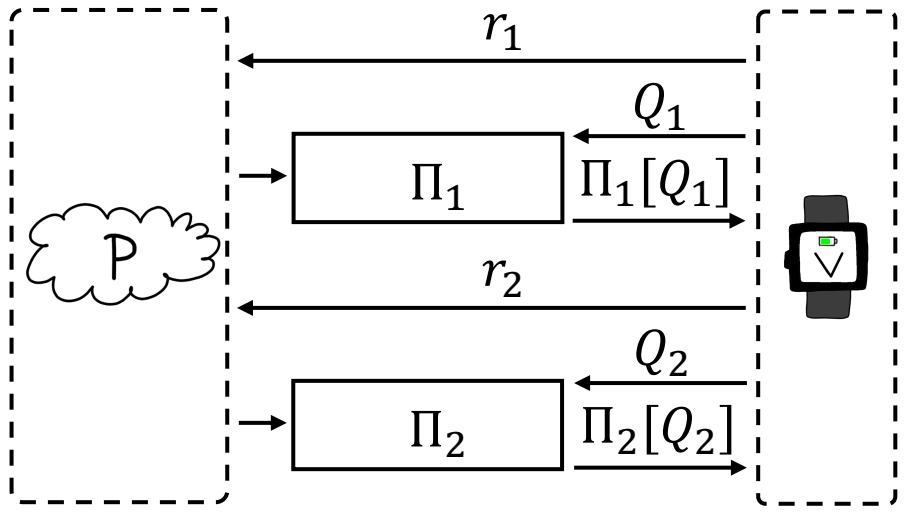
Proof strategy: Turn any 2-round IOP into 2-round FSKM-incompatible IOP (unsound for any h_{FS}, h_{CRHF})

Result: For any $\mathcal{L} \in NP$, there is an FSKM-incompatible 2-round IOP



Result: For any $\mathcal{L} \in NP$, there is an FSKM-incompatible 2-round IOP

2-Round IOP



V accepts if

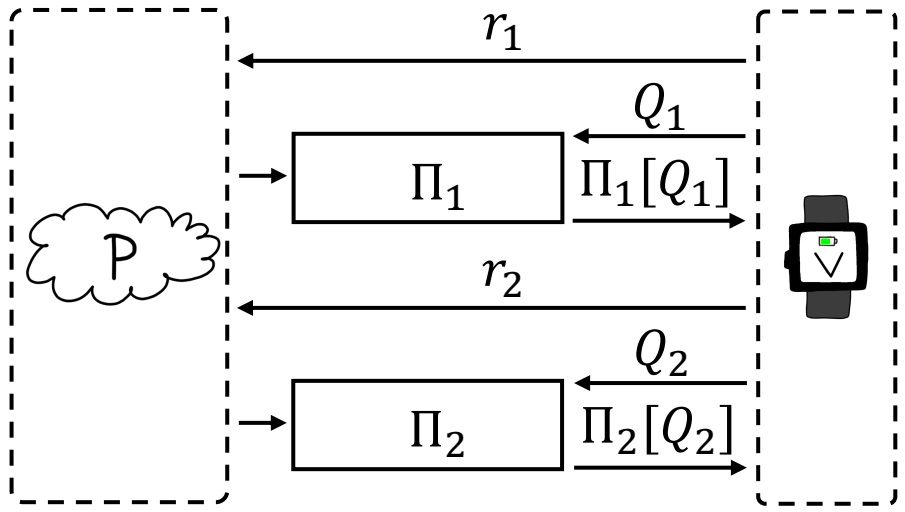
$V(r_1, r_2, Q_1, Q_2, \Pi_1[Q_1], \Pi_2[Q_2]) = 1$

FSKM-Incompatible IOP

Same interactive protocol, but V **also** accepts if Π_1 "predicts" next random message r_2.

Result: For any $\mathcal{L} \in NP$, there is an FSKM-incompatible 2-round IOP

2-Round IOP



accepts if

$V(r_1, r_2, Q_1, Q_2, \Pi_1[Q_1], \Pi_2[Q_2]) = 1$

FSKM-Incompatible IOP

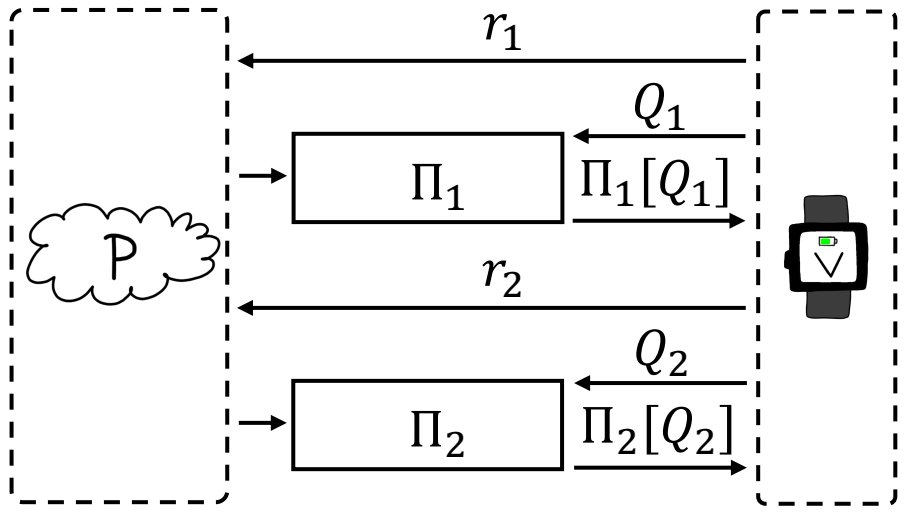
Same interactive protocol, but also accepts if Π_1 "predicts" next random message r_2.

Intuition: can cheat in FSKM since $r_2 = h_{FS}\left(\mathbf{Merkle}_{h_{CRHF}}(\Pi_1)\right)$ but original IOP still sound.

(Similar technique in [Barak 01, Goldwasser-Kalai 03])

Result: For any $\mathcal{L} \in NP$, there is an FSKM-incompatible 2-round IOP

2-Round IOP



accepts if

$V(r_1, r_2, Q_1, Q_2, \Pi_1[Q_1], \Pi_2[Q_2]) = 1$

FSKM-Incompatible IOP

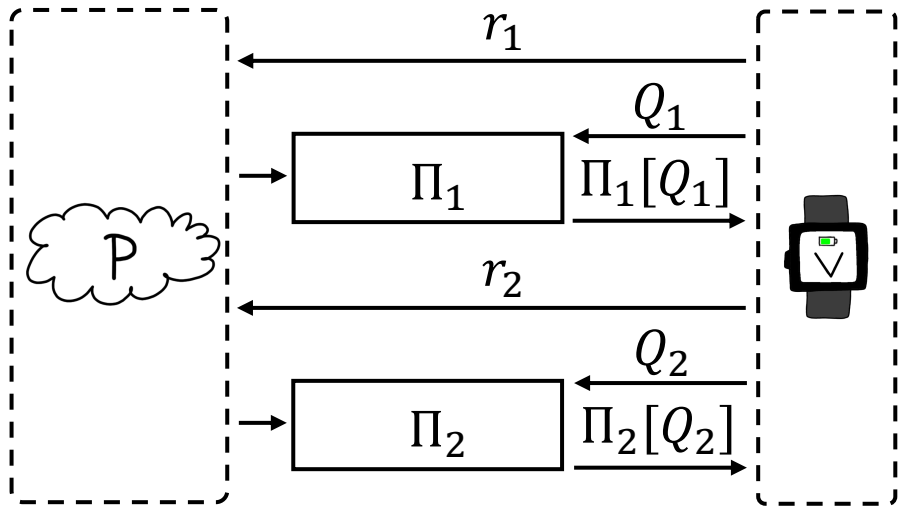
(Naive attempt)

Same interactive protocol, but also accepts if

- $\Pi_1 = \langle f \rangle, \langle g \rangle, \mathbf{0}$ where $f\left(\mathbf{Merkle}_g(\Pi_1)\right) = r_2$.

Result: For any $\mathcal{L} \in NP$, there is an FSKM-incompatible 2-round IOP

2-Round IOP



accepts if

$V(r_1, r_2, Q_1, Q_2, \Pi_1[Q_1], \Pi_2[Q_2]) = 1$

FSKM-Incompatible IOP
(Naive attempt)

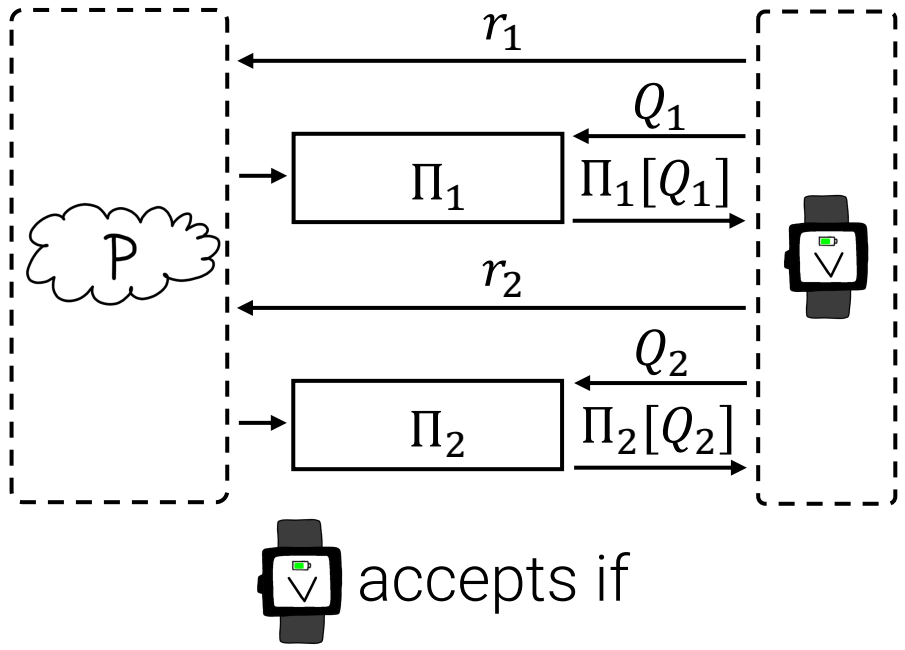
Same interactive protocol, but also accepts if

- $\Pi_1 = \langle f \rangle, \langle g \rangle, \mathbf{0}$ where $f\left(\mathbf{Merkle}_g(\Pi_1)\right) = r_2$.

But must read all of Π_1 to verify Merkle tree commitment.

Result: For any $\mathcal{L} \in NP$, there is an FSKM-incompatible 2-round IOP

2-Round IOP



accepts if

$V(r_1, r_2, Q_1, Q_2, \Pi_1[Q_1], \Pi_2[Q_2]) = 1$

FSKM-Incompatible IOP
(Second attempt)

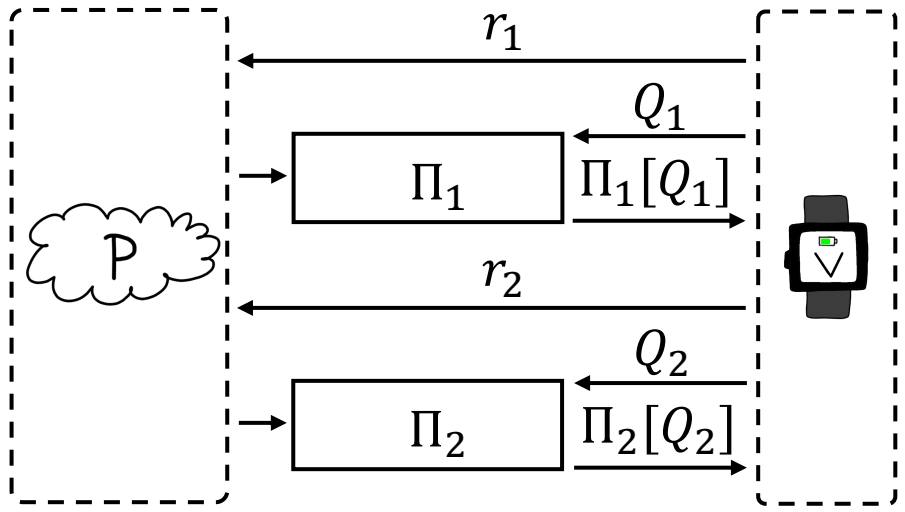
Same interactive protocol, but also accepts if

- Π_2 is PCP that (r_2, Π_1) satisfies

"$\Pi_1 = \langle f \rangle, \langle g \rangle, \mathbf{0}$ where
$f\left(\mathrm{Merkle}_g(\Pi_1)\right) = r_2$"

Result: For any $\mathcal{L} \in NP$, there is an FSKM-incompatible 2-round IOP

2-Round IOP



accepts if

$V(r_1, r_2, Q_1, Q_2, \Pi_1[Q_1], \Pi_2[Q_2]) = 1$

FSKM-Incompatible IOP
(Second attempt)

Same interactive protocol, but **also** accepts if

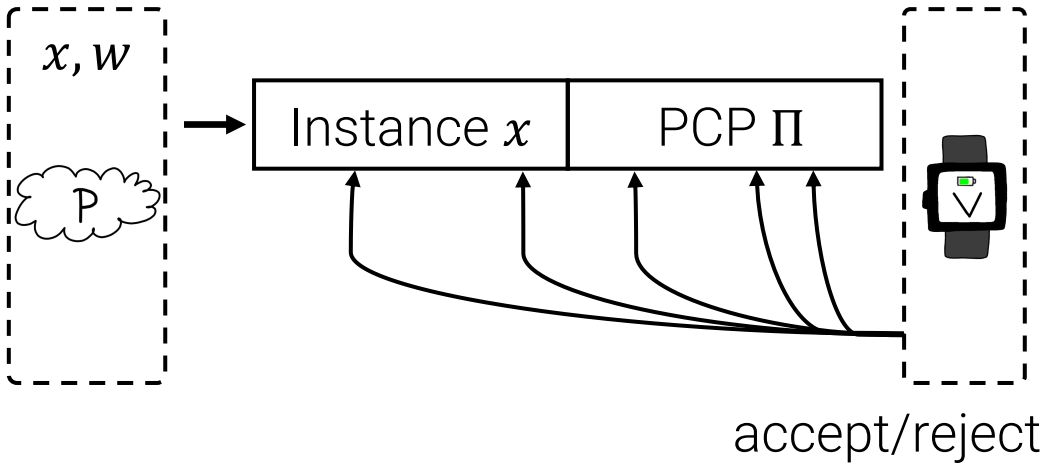
- Π_2 is PCP that (r_2, Π_1) satisfies

"$\Pi_1 = \langle f \rangle, \langle g \rangle, \mathbf{0}$ where
$f\left(\mathbf{Merkle}_g(\Pi_1)\right) = r_2$"

Problem: verifying PCP requires to know the instance.

Aside: PCPs of Proximity

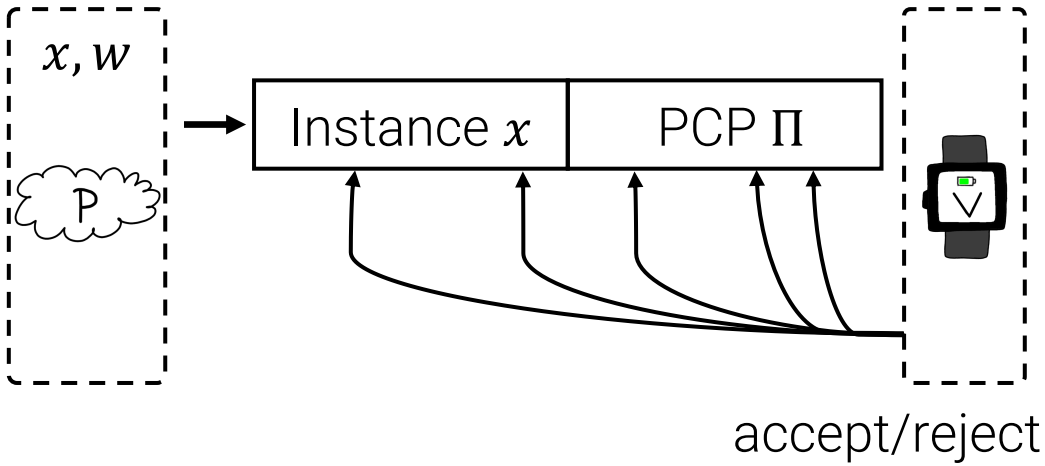
[Ben-Sasson-Goldreich-Harsha-Sudan-Vadhan 06, Dinur-Reingold 06, Dinur 07]



- Completeness: ⌚ accepts $x \in \mathcal{L}$
- Soundness: ⌚ rejects if x is "far" (hamming dist) from all $x' \in \mathcal{L}$

Aside: PCPs of Proximity

[Ben-Sasson-Goldreich-Harsha-Sudan-Vadhan 06, Dinur-Reingold 06, Dinur 07]



- Completeness: V accepts $x \in \mathcal{L}$
- Soundness: V rejects if x is "far" (hamming dist) from all $x' \in \mathcal{L}$

FSKM-Incompatible IOP

(Third attempt)

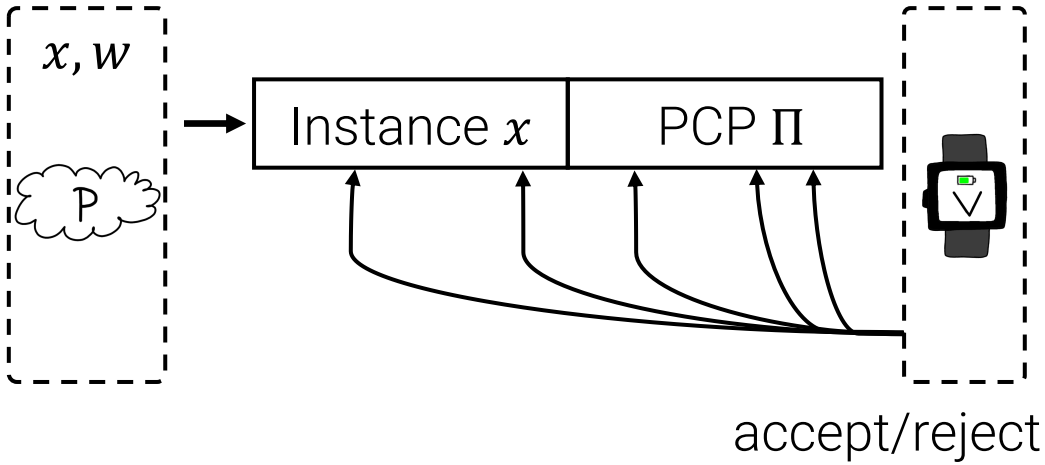
Same interactive protocol, but V also accepts if

- Π_2 is PCP of proximity that (r_2, Π_1) satisfies

  "$\Pi_1 = \langle f \rangle, \langle g \rangle$ where $f\left(\mathbf{Merkle}_g(\Pi_1)\right) = r_2$"

Aside: PCPs of Proximity

[Ben-Sasson-Goldreich-Harsha-Sudan-Vadhan 06, Dinur-Reingold 06, Dinur 07]



- Completeness: ⌚ accepts $x \in \mathcal{L}$
- Soundness: ⌚ rejects if x is "far" (hamming dist) from all $x' \in \mathcal{L}$

FSKM-Incompatible IOP

(Third attempt)

Same interactive protocol, but ⌚ also accepts if

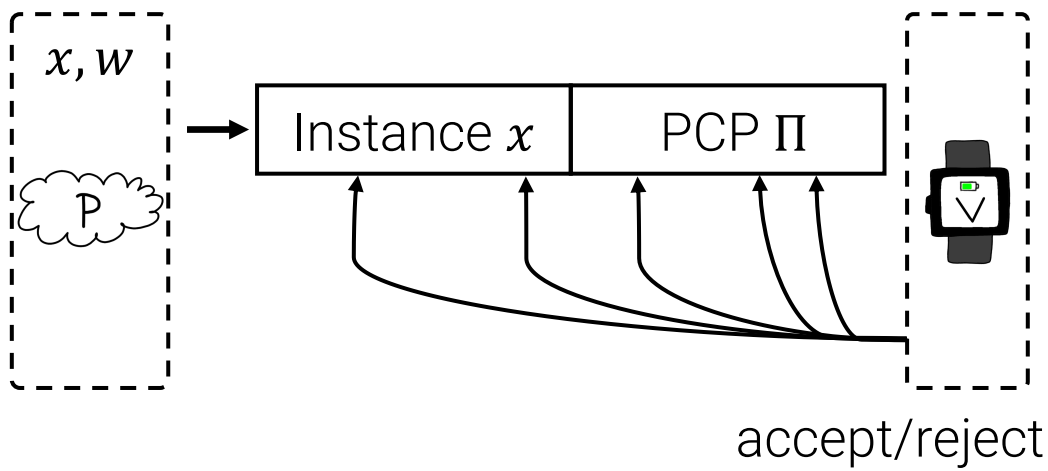
- Π_2 is PCP of proximity that (r_2, Π_1) satisfies

$$\text{“}\Pi_1 = \langle f \rangle, \langle g \rangle \text{ where } f\left(\text{Merkle}_g(\Pi_1)\right) = r_2\text{”}$$

IOP might not remain sound (maybe $\Pi_1 = \langle f \rangle, \langle g \rangle$ does not "predict" r_2, but some close $\Pi_1' = \langle f' \rangle, \langle g' \rangle$ does)

Aside: PCPs of Proximity

[Ben-Sasson-Goldreich-Harsha-Sudan-Vadhan 06, Dinur-Reingold 06, Dinur 07]



- Completeness: V accepts $x \in \mathcal{L}$
- Soundness: V rejects if x is "far" (hamming dist) from all $x' \in \mathcal{L}$

FSKM-Incompatible IOP

(final version)

Same interactive protocol, but V **also** accepts if

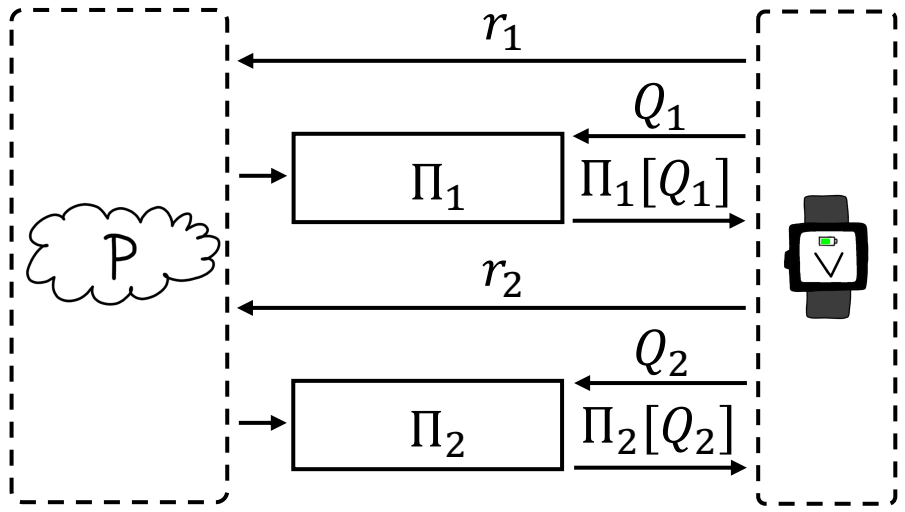
- Π_2 is PCP of proximity that (r_2, Π_1) satisfies

$$\text{“}\mathbf{Dec}(\Pi_1) = \langle f \rangle, \langle g \rangle \text{ where } f\left(\mathbf{Merkle}_g(\Pi_1)\right) = r_2\text{”}$$

(use error-correcting code $\mathbf{Enc}, \mathbf{Dec}$)

Result: There exists "bad" 2-round IOP s.t. FSKM is unsound for any h_{CRHF}, h_{FS}.

2-Round IOP



accepts if

$V(r_1, r_2, Q_1, Q_2, \Pi_1[Q_1], \Pi_2[Q_2]) = 1$

FSKM-Incompatible IOP
(final version)

Same interactive protocol, but also accepts if

- Π_2 is PCP of proximity that (r_2, Π_1) satisfies

"$\mathbf{Dec}(\Pi_1) = \langle f \rangle, \langle g \rangle$ where $f\left(\mathbf{Merkle}_g(\Pi_1)\right) = r_2$"

(use error-correcting code $\mathbf{Enc}, \mathbf{Dec}$)

Result: There exists "bad" 2-round IOP s.t. FSKM is unsound for any h_{CRHF}, h_{FS}.

New IOP retains soundness:

- If  accepts, then (r_2, Π_1) is close to accepting (r_2, Π_1')
- By error-correction, Π_1' and Π_1 encode same $\langle f \rangle, \langle g \rangle$.
- But r_2 sampled after Π_1!

FSKM-Incompatible IOP
(final version)

Same interactive protocol, but  **also** accepts if

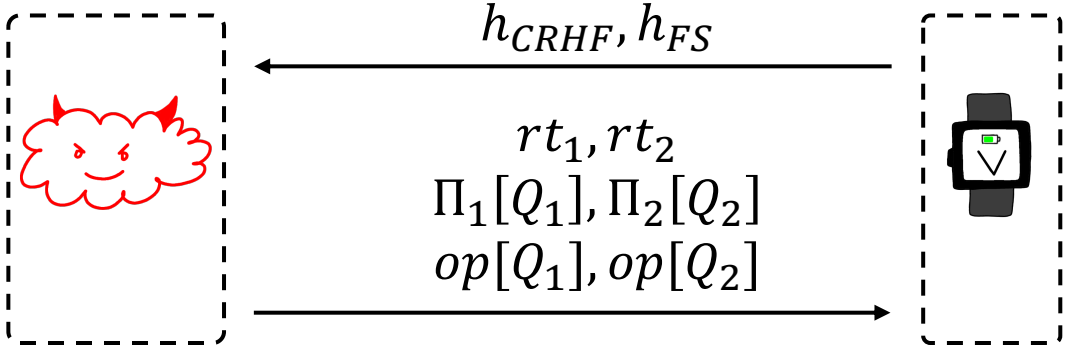
- Π_2 is PCP of proximity that (r_2, Π_1) satisfies

$$\text{"}\mathbf{Dec}(\Pi_1) = \langle f \rangle, \langle g \rangle \text{ where } f\left(\mathbf{Merkle}_g(\Pi_1)\right) = r_2\text{"}$$

(use error-correcting code $\mathbf{Enc}, \mathbf{Dec}$)

Result: There exists "bad" 2-round IOP s.t. FSKM is unsound for any h_{CRHF}, h_{FS}.

New IOP is unsound after FSKM



FSKM-Incompatible IOP
(final version)

Same interactive protocol, but **also** accepts if

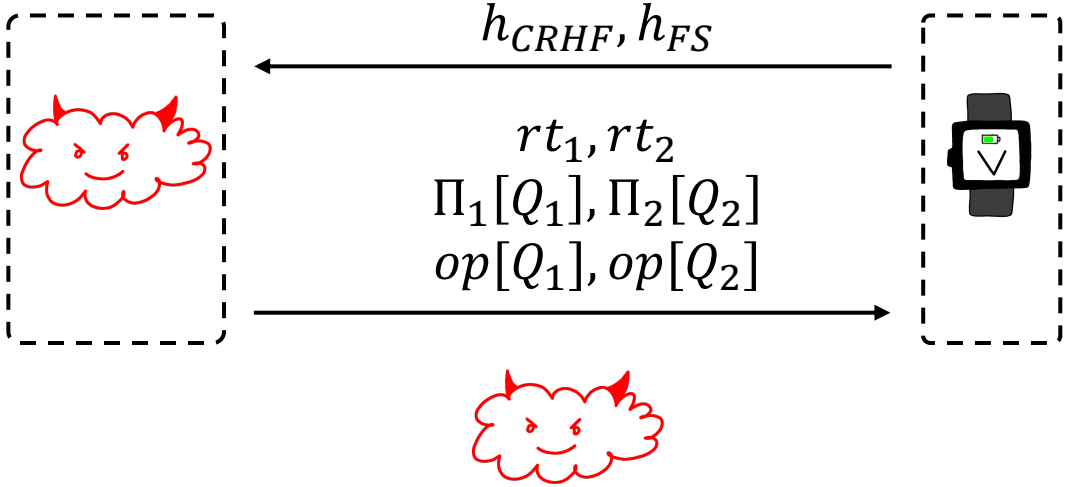
- Π_2 is PCP of proximity that (r_2, Π_1) satisfies

$$\text{“}\mathbf{Dec}(\Pi_1) = \langle f \rangle, \langle g \rangle \text{ where}$$
$$f\left(\mathbf{Merkle}_g(\Pi_1)\right) = r_2\text{”}$$

(use error-correcting code $\mathbf{Enc}, \mathbf{Dec}$)

Result: There exists "bad" 2-round IOP s.t. FSKM is unsound for any h_{CRHF}, h_{FS}.

New IOP is unsound after FSKM



1. Set $\Pi_1 = Enc(\langle h_{FS} \rangle, \langle h_{CRHF} \rangle)$.
2. Set $r_2 = h_{FS}\left(\mathrm{Merkle}_{h_{CRHF}}(\Pi_1)\right)$
3. PCPP Π_2 that (r_2, Π_1) satisfies

will *always* accept

FSKM-Incompatible IOP
(final version)

Same interactive protocol, but **also** accepts if

- Π_2 is PCP of proximity that (r_2, Π_1) satisfies

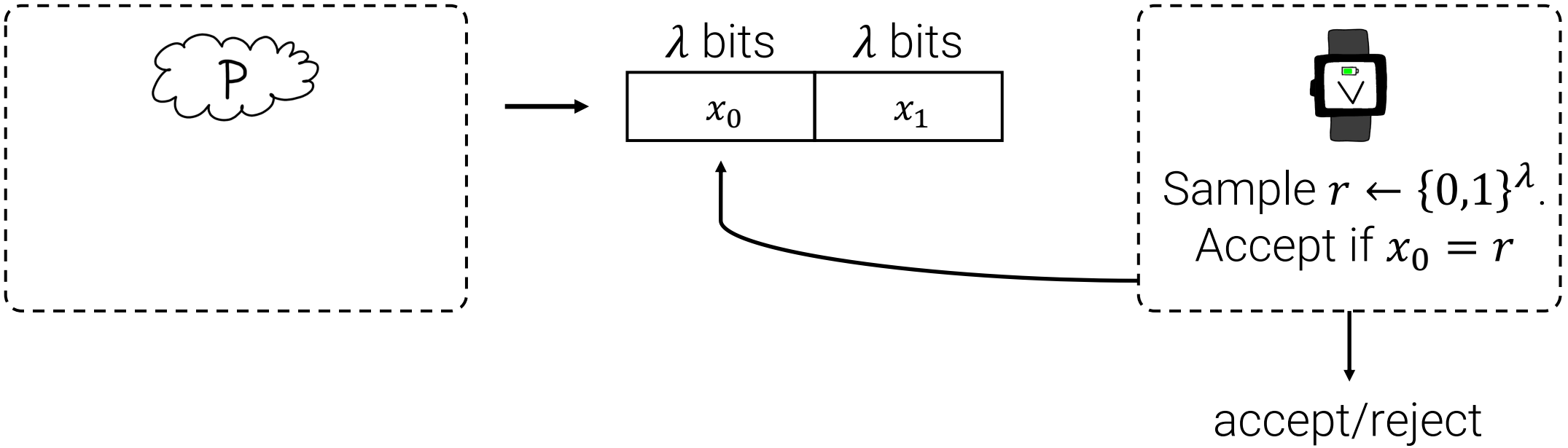
"$\mathrm{Dec}(\Pi_1) = \langle f \rangle, \langle g \rangle$ where $f\left(\mathrm{Merkle}_g(\Pi_1)\right) = r_2$"

(use error-correcting code $\mathbf{Enc}, \mathbf{Dec}$)

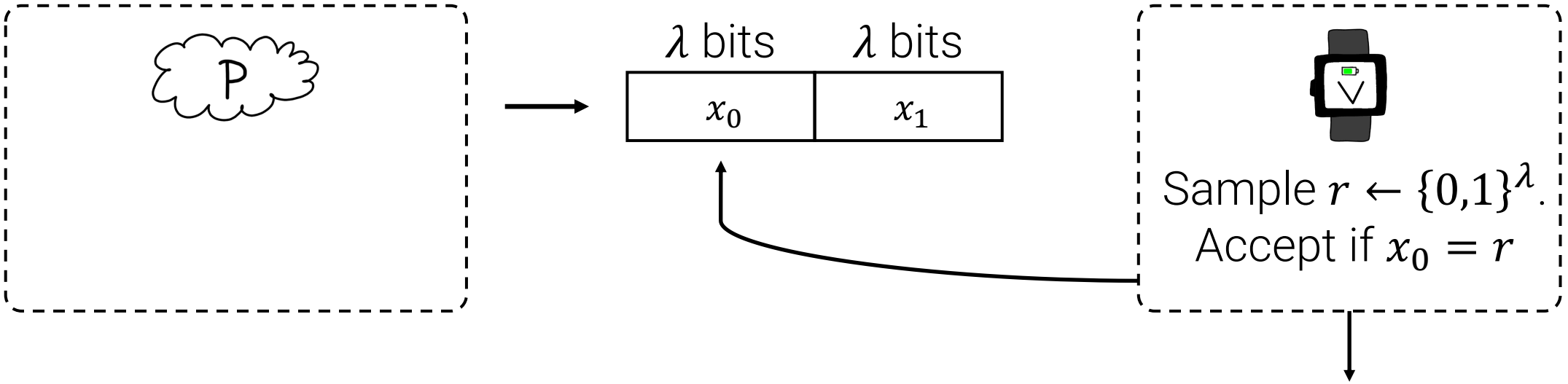
Outline

- (Review) Kilian's Protocol
- A Bad IOP for IOP-Based SNARGs
- (If time) A Bad CRHF for PCP-Based SNARGs

A Silly PCP for the Empty Language

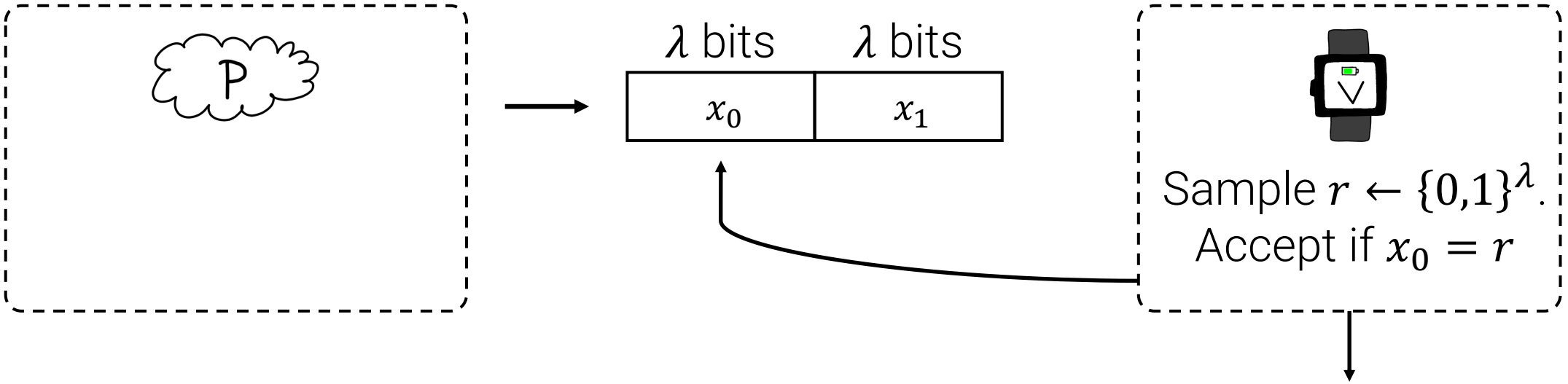


A Silly PCP for the Empty Language



Today: Construct h^*_{CRHF} s.t. FSKM is always insecure for this PCP for any h_{FS}.

A Silly PCP for the Empty Language

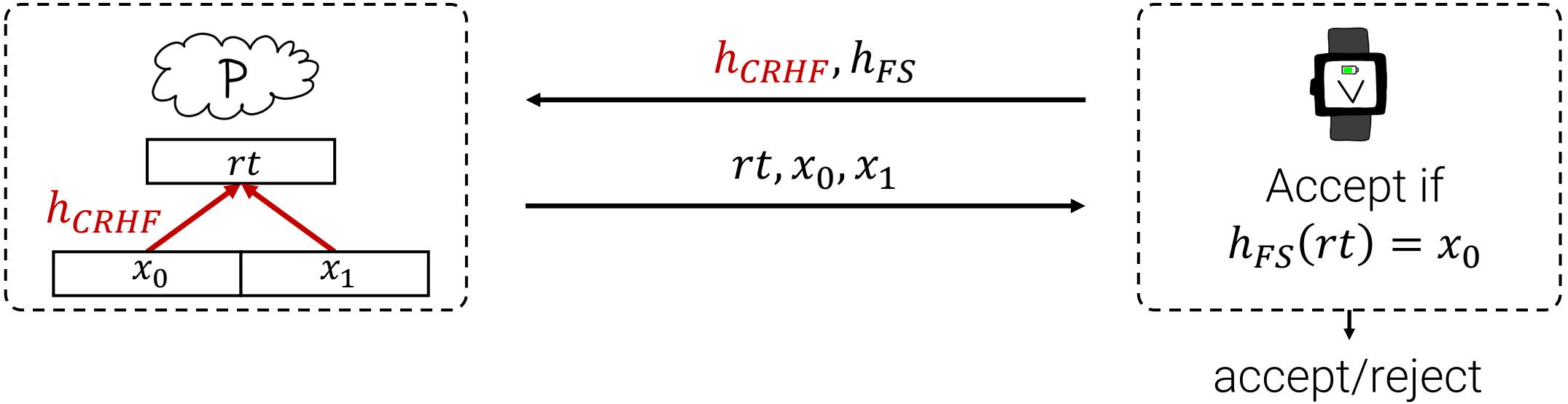


Today: Construct h^*_{CRHF} s.t. FSKM is always insecure for this PCP for any h_{FS}.

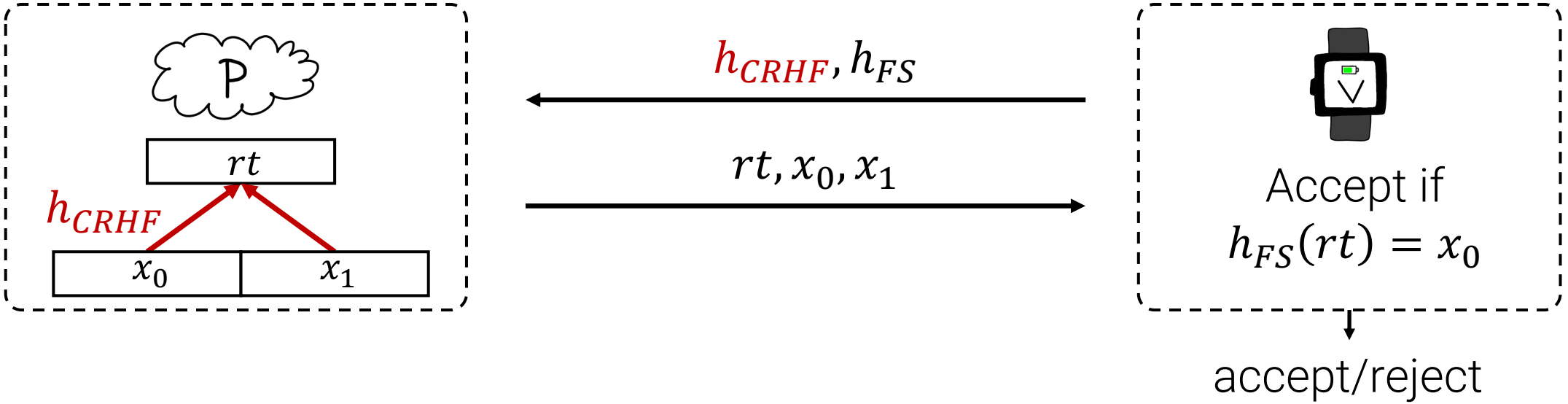
In the paper: Construct h^*_{CRHF} s.t. FSKM is always insecure for any** PCP for any h_{FS}.

**For any PCP that does not already yield a SNARG

FSKM for Empty Language PCP



FSKM for Empty Language PCP

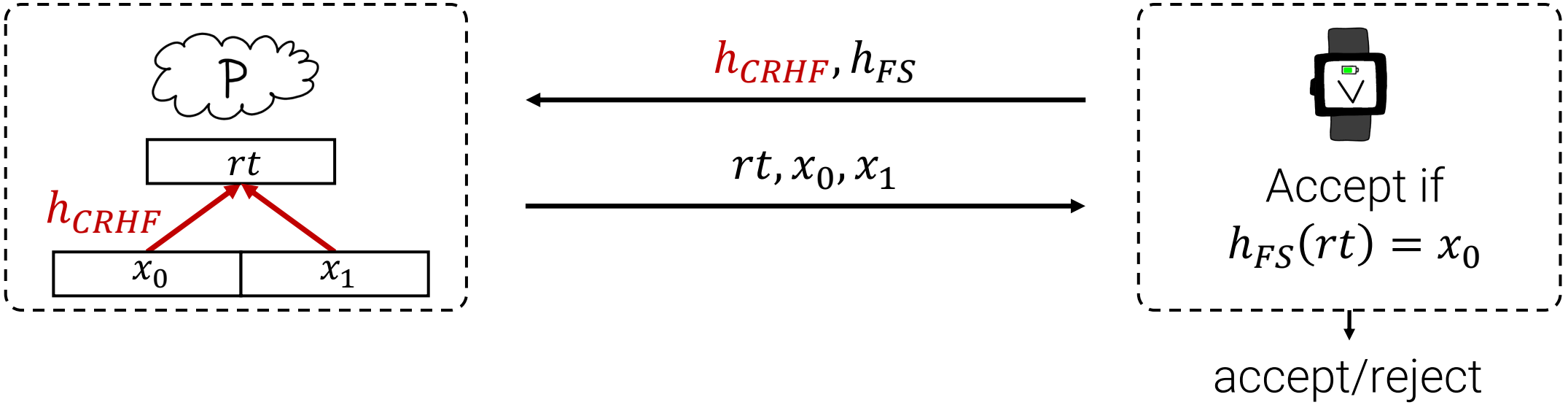


Cheating finds x_0, x_1 s.t.

$h_{CRHF}(x_0, x_1) = rt$ and $h_{FS}(rt) = x_0$.

Strategy: Build h_{CRHF} to enable this

FSKM for Empty Language PCP



Cheating finds x_0, x_1 s.t.

$h_{CRHF}(x_0, x_1) = rt$ and $h_{FS}(rt) = x_0$.

Strategy: Build h_{CRHF} to enable this

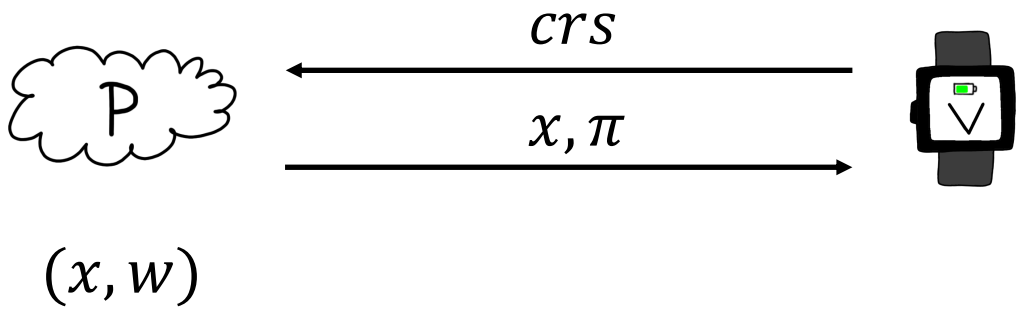
h_{CRHF} ***circularly-tractable*** if given $\langle f \rangle$, can efficiently generate x_0, x_1 where

$$z = h_{CRHF}(x_0, x_1)$$

$$x_0 = f(z)$$

Aside: SNARKs for NP

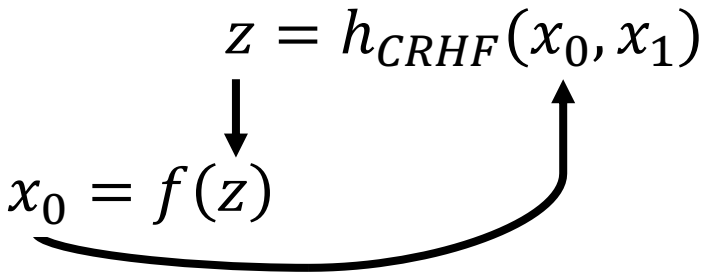
[Bitansky-Canetti-Chiesa-Tromer 11]



Very roughly: SNARK = SNARG + Knowledge

- **Adaptive Proof of Knowledge:** If P convinces V to accept x, there is efficient $\mathcal{E}_{\text{P}}$ outputting (x, w).

h^*_{CRHF} *circularly-tractable* if given $\langle f \rangle$, can efficiently generate x_0, x_1 where



Constructing h^*_{CRHF}:

$\{0,1\}^{2\lambda} \rightarrow \{0,1\}^{\lambda}$

h^*_{CRHF} hash key

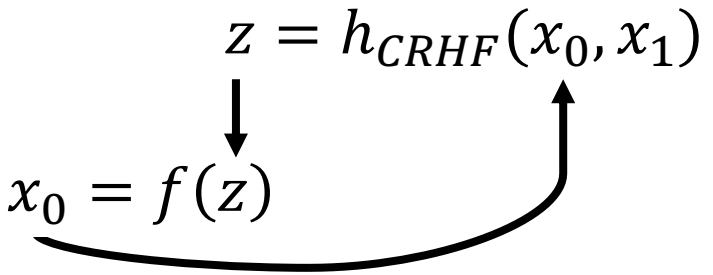
- Two CRHFs:

$h_1: \{0,1\}^{2\lambda} \rightarrow \{0,1\}^{\lambda}$

$h_2: \{0,1\}^{*} \rightarrow \{0,1\}^{\lambda/2}$

- SNARK crs

h^*_{CRHF} *circularly-tractable* if given $\langle f \rangle$, can efficiently generate x_0, x_1 where



Constructing h^*_{CRHF}:

$\{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$

h^*_{CRHF} hash key

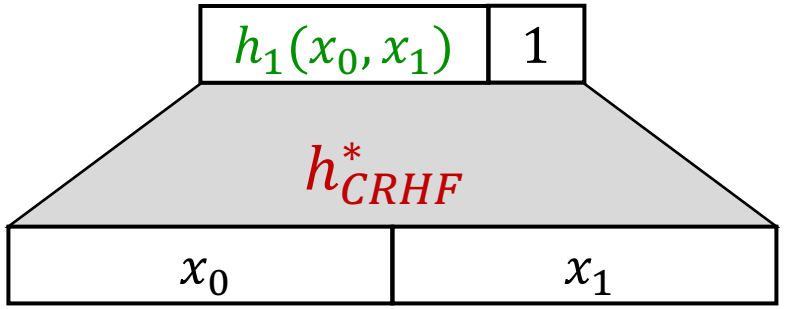
- Two CRHFs:

$h_1: \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$

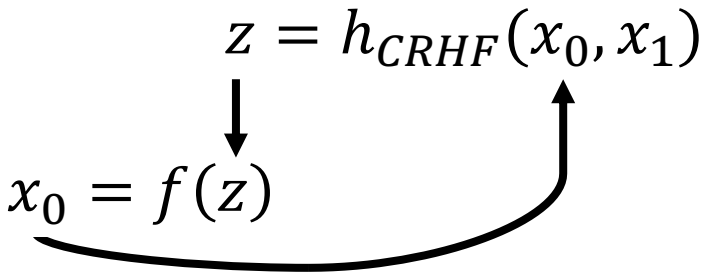
$h_2: \{0,1\}^{*} \to \{0,1\}^{\lambda/2}$

- SNARK crs

"Normal" inputs:



h^*_{CRHF} *circularly-tractable* if given $\langle f \rangle$, can efficiently generate x_0, x_1 where

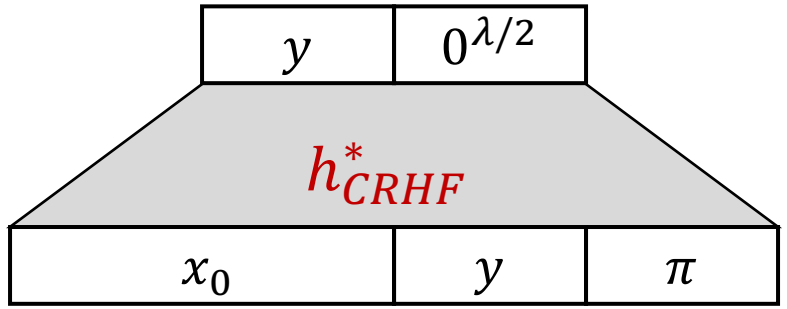


Constructing h^*_{CRHF}:
$\{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$

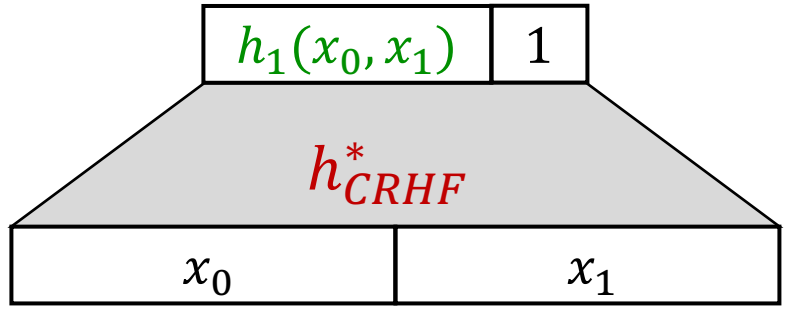
h^*_{CRHF} hash key

- Two CRHFs:
  $h_1: \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$
  $h_2: \{0,1\}^{*} \to \{0,1\}^{\lambda/2}$
- SNARK crs

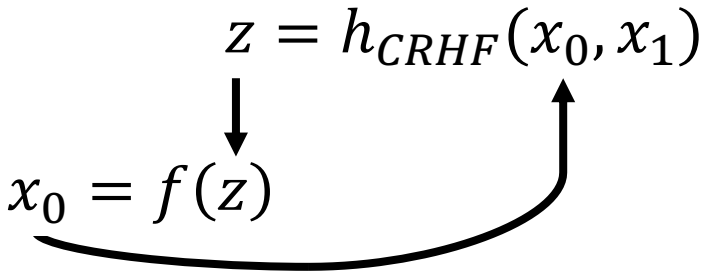
"Special" inputs:
(x_0, y, π) s.t. π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f(y, 0^{\lambda/2})$



"Normal" inputs:
everything else



h^*_{CRHF} *circularly-tractable* if given $\langle f \rangle$, can efficiently generate x_0, x_1 where

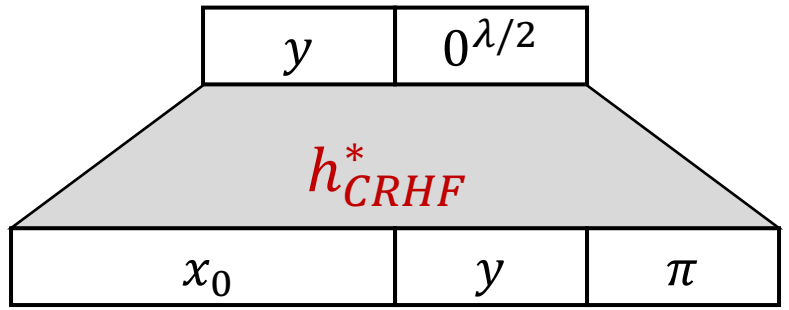


h^*_{CRHF} is circularly-tractable

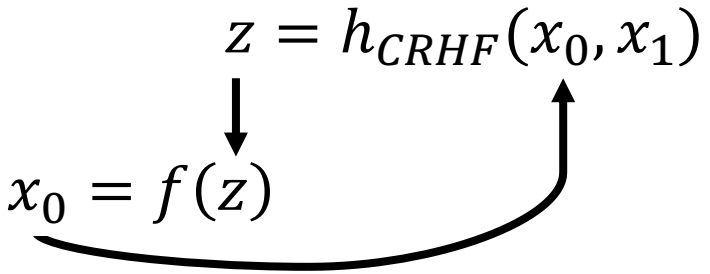
- Two CRHFs:
  $h_1: \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$
  $h_2: \{0,1\}^{*} \to \{0,1\}^{\lambda/2}$
- SNARK crs

h^*_{CRHF} hash key

"Special" inputs:
(x_0, y, π) s.t. π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f(y, 0^{\lambda/2})$



h^*_{CRHF} *circularly-tractable* if given $\langle f \rangle$, can efficiently generate x_0, x_1 where



h^*_{CRHF} is circularly-tractable

- Given $\langle f \rangle$, compute $h_2(\langle f \rangle) = y$
- Let $x_0 = f(y, 0^\lambda)$

$$x_0 = f\left(y, 0^\lambda\right)$$

- Two CRHFs:

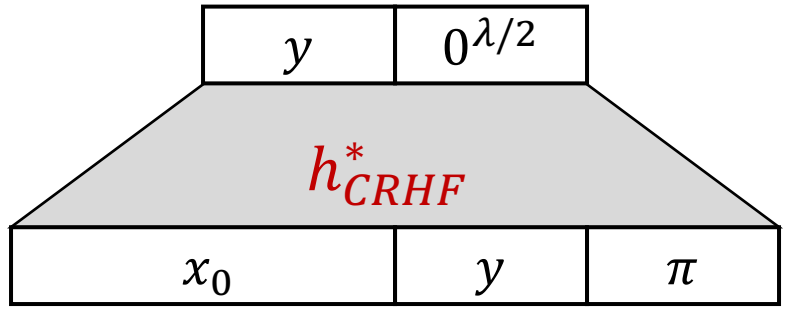
$h_1 : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$

$h_2 : \{0,1\}^* \to \{0,1\}^{\lambda/2}$

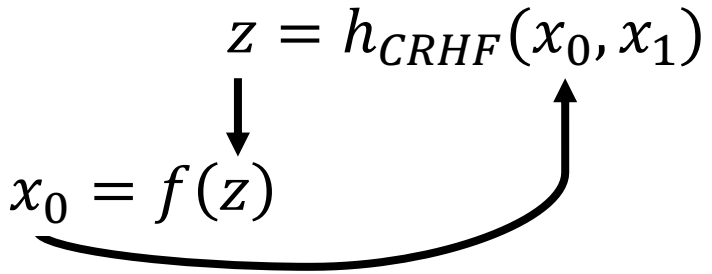
- SNARK crs

h^*_{CRHF} hash key

"Special" inputs:
(x_0, y, π) s.t. π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f\left(y, 0^{\lambda/2}\right)$

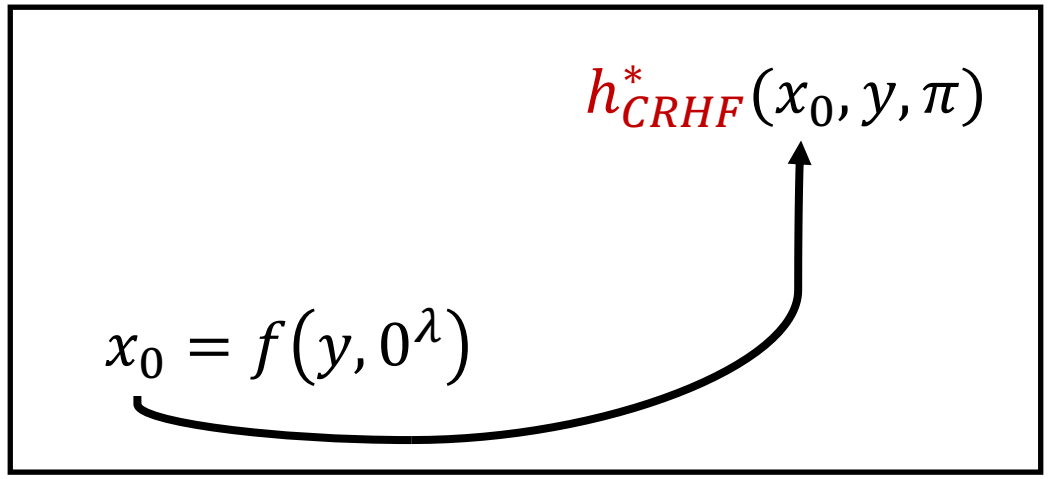


h^*_{CRHF} *circularly-tractable* if given $\langle f \rangle$, can efficiently generate x_0, x_1 where



h^*_{CRHF} is circularly-tractable

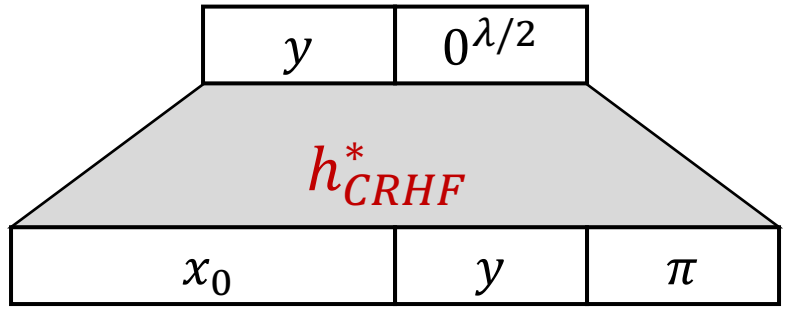
- Given $\langle f \rangle$, compute $h_2(\langle f \rangle) = y$
- Let $x_0 = f(y, 0^\lambda)$
- Use $\langle f \rangle$ to generate SNARK π
- Let $x_1 = y, \pi$



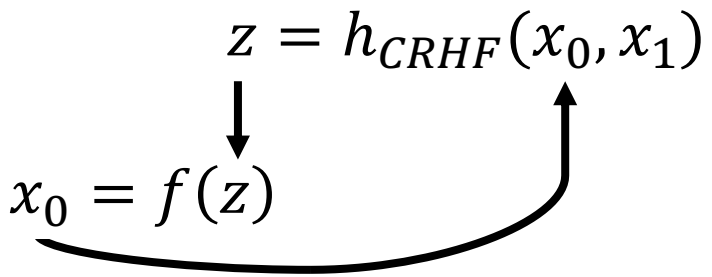
- Two CRHFs:
  $h_1: \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$
  $h_2: \{0,1\}^* \to \{0,1\}^{\lambda/2}$
- SNARK crs

h^*_{CRHF} hash key

"Special" inputs:
(x_0, y, π) s.t. π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f(y, 0^{\lambda/2})$

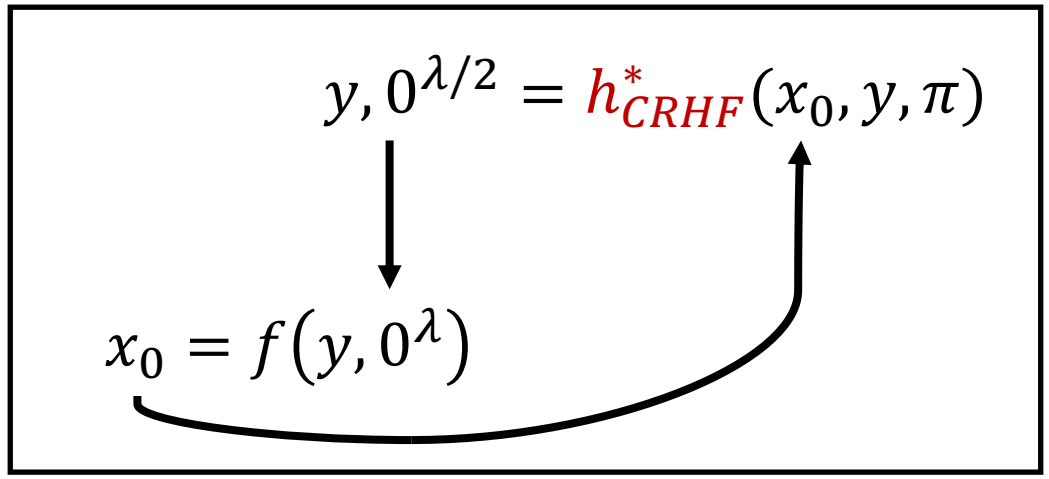


h^*_{CRHF} ***circularly-tractable*** if given $\langle f \rangle$, can efficiently generate x_0, x_1 where



h^*_{CRHF} is circularly-tractable

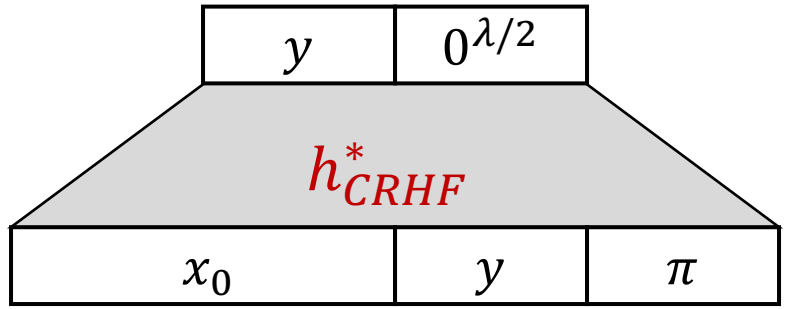
- Given $\langle f \rangle$, compute $h_2(\langle f \rangle) = y$
- Let $x_0 = f(y, 0^\lambda)$
- Use $\langle f \rangle$ to generate SNARK π
- Let $x_1 = y, \pi$



- Two CRHFs:
  $h_1: \{0,1\}^{2\lambda} \rightarrow \{0,1\}^\lambda$
  $h_2: \{0,1\}^* \rightarrow \{0,1\}^{\lambda/2}$
- SNARK crs

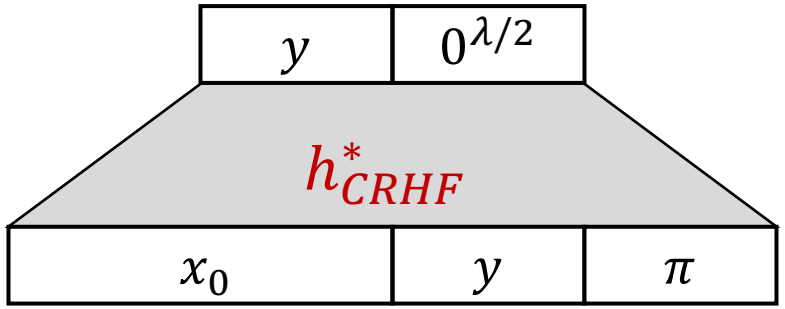
h^*_{CRHF} hash key

"Special" inputs:
(x_0, y, π) s.t. π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f(y, 0^{\lambda/2})$



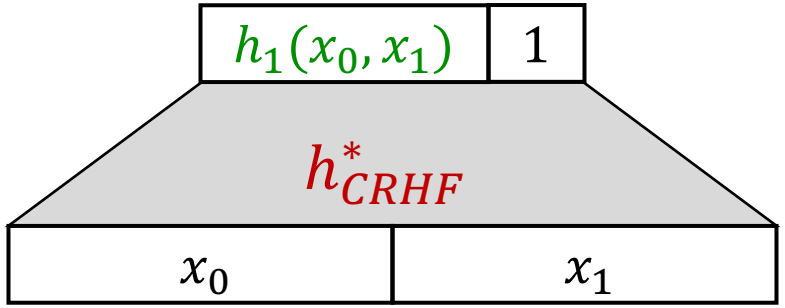
"Special" inputs:

(x_0, y, π) where π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f\left(y, 0^{\lambda/2}\right)$



"Normal" inputs:

everything else



- Two CRHFs:

$h_1: \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$

$h_2: \{0,1\}^{*} \to \{0,1\}^{\lambda/2}$

- SNARK crs

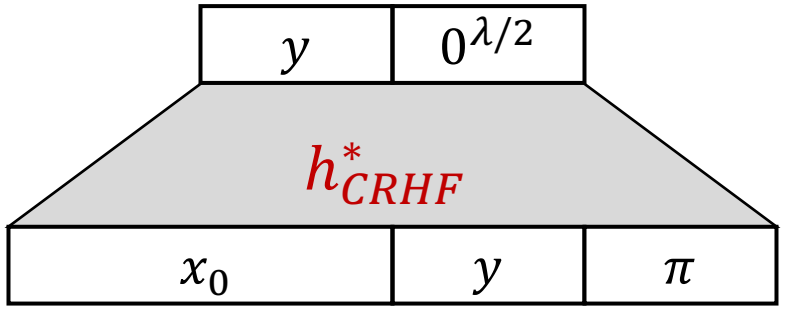
h^*_{CRHF} hash key

Collision resistance?

Only need to check "special" inputs

"Special" inputs:

(x_0, y, π) where π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f\left(y, 0^{\lambda/2}\right)$



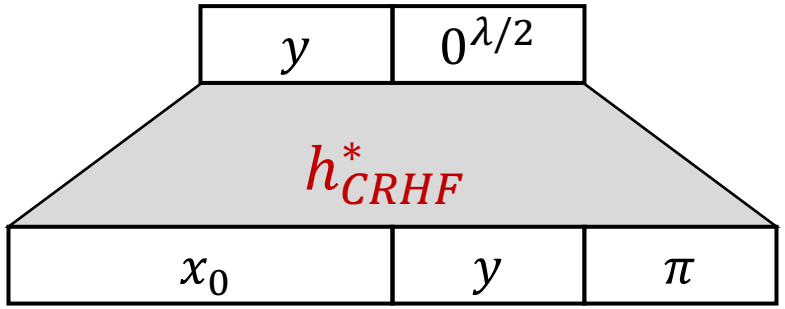
Collision must be of form $(x_0, y, \pi), (x_0', y, \pi')$

If $x_0 \neq x_0'$:

- π extracts to $\langle f \rangle$, π' extracts to $\langle f' \rangle$

"Special" inputs:

(x_0, y, π) where π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f\left(y, 0^{\lambda/2}\right)$



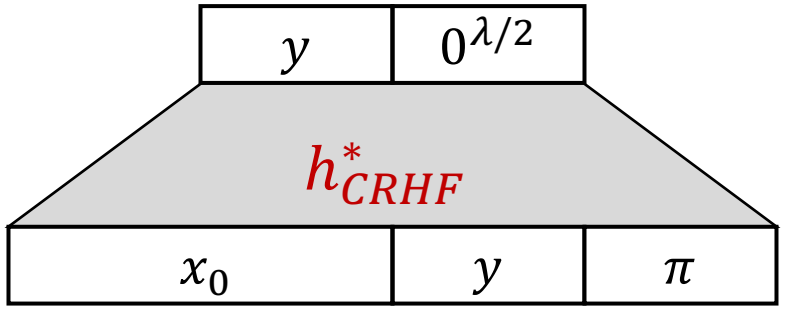
Collision must be of form $(x_0, y, \pi), (x_0', y, \pi')$

If $x_0 \neq x_0'$:

- π extracts to $\langle f \rangle$, π' extracts to $\langle f' \rangle$
- $f\left(y, 0^{\lambda/2}\right) = x_0$ and $f'\left(y, 0^{\lambda/2}\right) = x_0{}'$.

"Special" inputs:

(x_0, y, π) where π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f\left(y, 0^{\lambda/2}\right)$



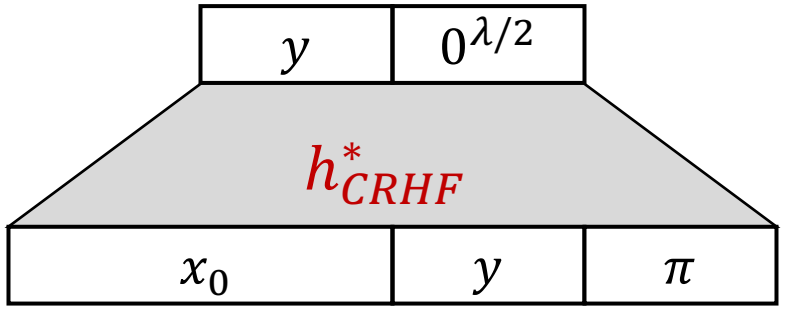
Collision must be of form $(x_0, y, \pi), (x_0', y, \pi')$

If $x_0 \neq x_0'$:

- π extracts to $\langle f \rangle$, π' extracts to $\langle f' \rangle$
- $f\left(y, 0^{\lambda/2}\right) = x_0$ and $f'\left(y, 0^{\lambda/2}\right) = x_0{}'$.
- implies $\langle f \rangle \neq \langle f' \rangle$ (collision for h_2)

"Special" inputs:

(x_0, y, π) where π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f\left(y, 0^{\lambda/2}\right)$



Collision must be of form $(x_0, y, \pi), (x_0', y, \pi')$

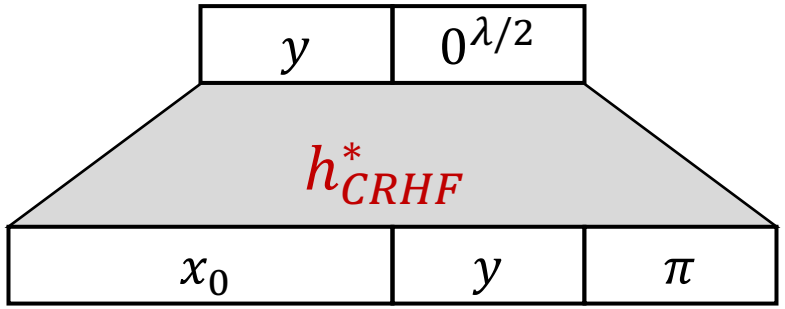
If $x_0 \neq x_0'$:

- π extracts to $\langle f \rangle$, π' extracts to $\langle f' \rangle$
- $f\left(y, 0^{\lambda/2}\right) = x_0$ and $f'\left(y, 0^{\lambda/2}\right) = x_0{}'$.
- implies $\langle f \rangle \neq \langle f' \rangle$ (collision for h_2)

So $x_0 = x_0'$, but $\pi \neq \pi'$

"Special" inputs:

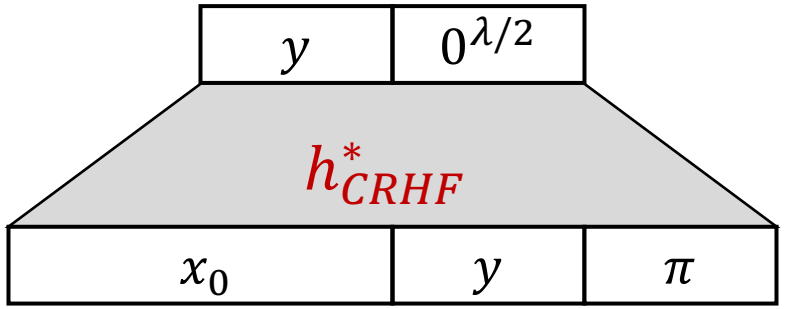
(x_0, y, π) where π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f\left(y, 0^{\lambda/2}\right)$



So collision must be $(x_0, y, \pi), (x_0, y, \pi')$ for $\pi \neq \pi'$

"Special" inputs:

(x_0, y, π) where π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f\left(y, 0^{\lambda/2}\right)$

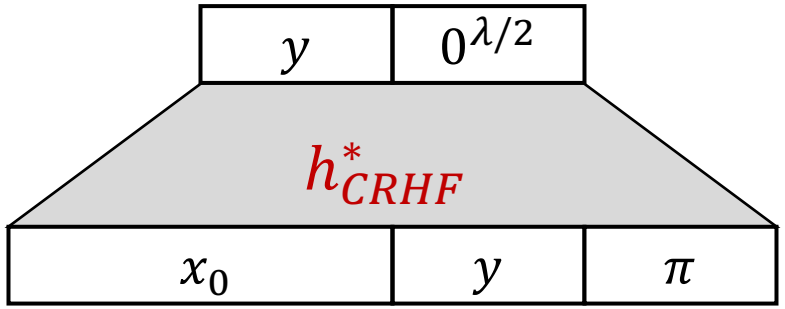


So collision must be $(x_0, y, \pi), (x_0, y, \pi')$ for $\pi \neq \pi'$

"Computationally-Unique" SNARK

If $\pi \neq \pi'$, then $\mathcal{E}_{\text{P}}$ can extract **distinct witnesses** $\langle f \rangle, \langle f' \rangle$.

"Special" inputs:
(x_0, y, π) where π is SNARK of $\langle f \rangle$ satisfying $h_2(\langle f \rangle) = y$ and $x_0 = f\left(y, 0^{\lambda/2}\right)$



So collision must be $(x_0, y, \pi), (x_0, y, \pi')$ for $\pi \neq \pi'$

"Computationally-Unique" SNARK

If $\pi \neq \pi'$, then $\mathcal{E}_{\text{P}}$ can extract **distinct witnesses** $\langle f \rangle, \langle f' \rangle$.

Yields collision for h_2!

Thank You!

Questions?

Slide Artwork by Eysa Lee