Witness-Succinct Universally-Composable SNARKs

Chaya Ganesh

Yashvanth Kondi

Claudio Orlandi

Mahak Pancholi

Akira Takahashi

Daniel Tschudi

Indian Institute of Science
भारतीय विज्ञान संस्थान

AARHUS UNIVERSITY

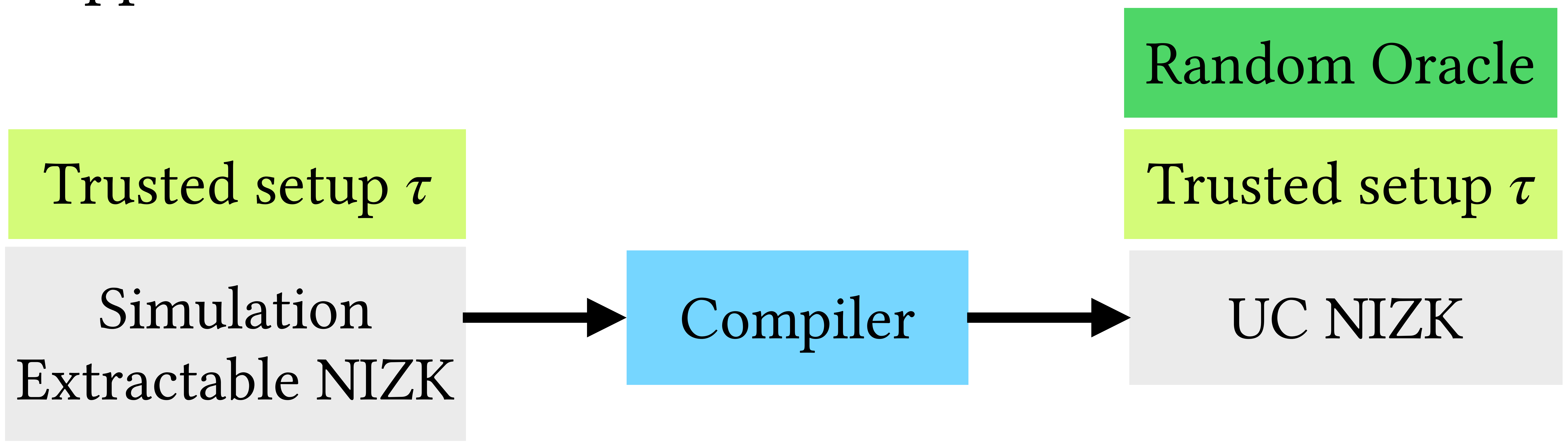
CONCORDIUM

To appear at Eurocrypt 2023

In a Nutshell

- We present the first constant-sized Universally Composable (UC) Non-interactive Zero-Knowledge Proofs

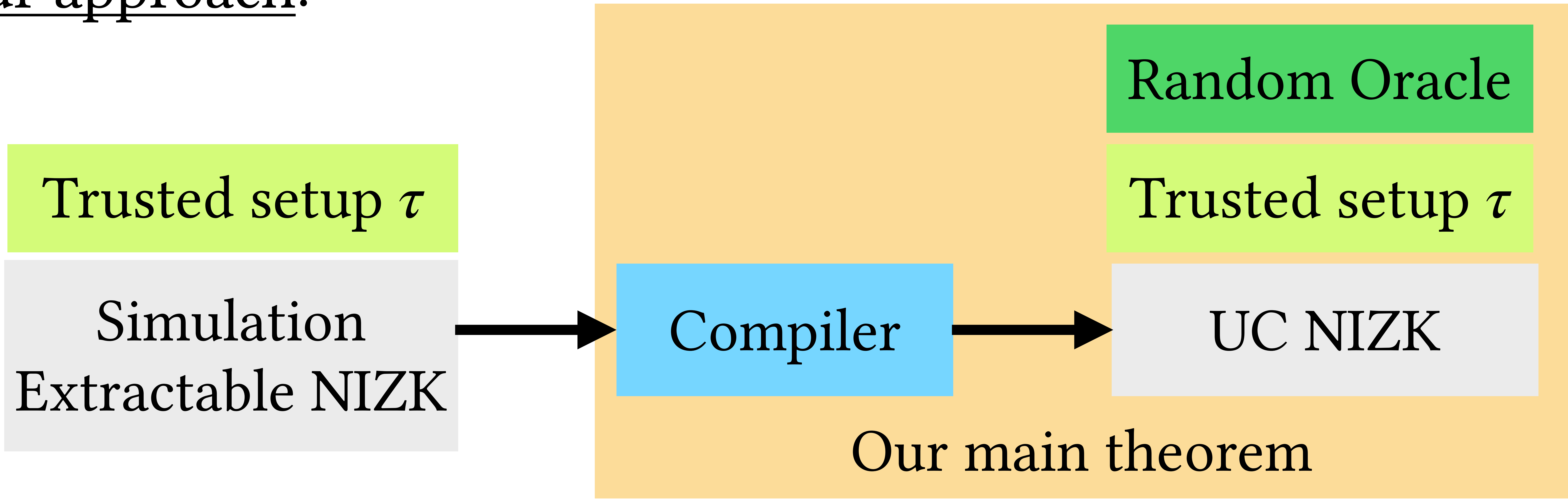
- <u>Our approach</u>:



In a Nutshell

- We present the first constant-sized Universally Composable (UC) Non-interactive Zero-Knowledge Proofs

- Our approach:



Witness-Succinct Universally Composable SNARKs

SNARKs

- Type of cryptographic proof
  - **S**uccinct — proof size is smaller than circuit or witness
  - **N**on-interactive — single message
  - **Ar**gument of **K**nowledge — witness is "extractable" from prover
- Many constructions, with tradeoffs in proof size, prover running time, verification cost, trusted setup, security guarantee
- **This talk**: focus on best possible succinctness—$O_\kappa(1)$ sized proofs

SNARKs

- Type of cryptographic proof
  - **S**uccinct — proof size is smaller than circuit or witness
  - **N**on-interactive — single message
  - **Ar**gument of **K**nowledge — witness is "extractable" from prover
- Many constructions, with tradeoffs in proof size, prover running time, verification cost, trusted setup, security guarantee
- **This talk**: focus on best possible succinctness—$O_\kappa(1)$ sized proofs

Security parameter terms are constants

Universally Composable

- Framework for concurrent security introduced in [Canetti 01]
- Guarantees composition in any context
- Modular, convenient to work with as a protocol designer
- ...but is challenging to achieve

Witness-Succinct

- *Witness* succinctness: proof size $|\pi| \in O_\kappa(1)$
- Contrast with *circuit* succinctness: $|\pi| = \theta_\kappa(|w|) + o_\kappa(|C|)$

Witness-Succinct

- *Witness* succinctness: proof size $|\pi| \in O_\kappa(1)$
- Contrast with *circuit* succinctness: $|\pi| = \theta_\kappa(|w|) + o_\kappa(|C|)$

Not a problem when witness is small
But imagine proving statements about a large pre-image of a public digest, etc.

This Talk

- **What will be covered**:
  Technique to lift Simulation Extractable (SE) SNARKS to UC security at $O_\kappa(1)$ overhead

- **What won't be touched**:
  How to instantiate SE SNARKs, intricacies and formalism of UC (this is to help understanding, not to hand-wave; please ask if something is unclear!)

Structure of this talk

1. Quick recap: NIZK | Why UC?
2. What existing works already achieve | What makes achieving UC difficult | A (too) simple approach
3. Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments
4. Final remarks

Structure of this talk

1 Quick recap: NIZK | Why UC?

2 What existing works already achieve | What makes achieving UC difficult | A (too) simple approach

3 Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments

4 Final remarks

Recap: NIZK

$$P(x, w) \xrightarrow{\quad\quad \pi \quad\quad} V(x)$$

- **Completeness**: An honest proof always verifies
- **Zero-knowledge**: π is simulatable without a witness (but a special trapdoor)
- **Argument of knowledge**: w can be extracted from π when $V(x, \pi) = 1$

Recap: NIZK

$P(x, w)$ Trusted setup τ $V(x)$

$$\xrightarrow{\pi}$$

- **Completeness**: An honest proof always verifies
- **Zero-knowledge**: π is simulatable without a witness (but a special trapdoor)
- **Argument of knowledge**: w can be extracted from π when $V(x, \pi) = 1$

Recap: NIZK

$\mathsf{Sim}(\mathsf{td}, x)$ Trusted setup τ

$V(x)$

$\xrightarrow{\tilde{\pi}}$

- **Completeness**: An honest proof always verifies
- **Zero-knowledge**: π is simulatable without a witness (but a special trapdoor)
- **Argument of knowledge**: w can be extracted from π when $V(x, \pi) = 1$

Recap: NIZK

$\mathsf{Sim}(\boxed{\mathsf{td}}, x)$ Trusted setup $\boxed{\tau}$

$V(x)$

$$\xrightarrow{\tilde{\pi}}$$

- **Completeness**: An honest proof always verifies
- **Zero-knowledge**: π is simulatable without a witness (but a special trapdoor)
- **Argument of knowledge**: w can be extracted from π when $V(x, \pi) = 1$

Recap: NIZK

$\mathsf{Sim}(\boxed{\mathsf{td}}, x)$ Trusted setup $\boxed{\tau}$

$V(x)$

$$\xrightarrow[\approx\ \pi]{\tilde{\pi}}$$

- **Completeness**: An honest proof always verifies
- **Zero-knowledge**: π is simulatable without a witness (but a special trapdoor)
- **Argument of knowledge**: w can be extracted from π when $V(x, \pi) = 1$

Recap: NIZK

$$P(x, w) \quad \boxed{\text{Trusted setup } \tau} \quad V(x)$$

$$\xrightarrow{\quad\quad \pi \quad\quad}$$

- **Completeness**: An honest proof always verifies
- **Zero-knowledge**: π is simulatable without a witness (but a special trapdoor)
- **Argument of knowledge**: w can be extracted from π when $V(x, \pi) = 1$

Recap: NIZK

$P(x, w)$

Trusted setup τ $\mathsf{Ext}(\mathsf{td}, x)$

$\xrightarrow{\pi}$

- **Completeness**: An honest proof always verifies
- **Zero-knowledge**: π is simulatable without a witness (but a special trapdoor)
- **Argument of knowledge**: w can be extracted from π when $V(x, \pi) = 1$

Recap: NIZK

$P(x, w)$ Trusted setup $\boxed{\tau}$ $\mathsf{Ext}(\boxed{\mathsf{td}}, x)$

$$\xrightarrow{\pi}$$

- **Completeness**: An honest proof always verifies
- **Zero-knowledge**: π is simulatable without a witness (but a special trapdoor)
- **Argument of knowledge**: w can be extracted from π when $V(x, \pi) = 1$

Recap: NIZK

$P(x, w)$

Trusted setup $\boxed{\tau}$ Ext($\boxed{\mathsf{td}}$, x)

$\xrightarrow{\pi}$

Output w

- **Completeness**: An honest proof always verifies
- **Zero-knowledge**: π is simulatable without a witness (but a special trapdoor)
- **Argument of knowledge**: w can be extracted from π when $V(x, \pi) = 1$

Structure of this talk

1	Quick recap: NIZK	Why UC?	
2	What existing works already achieve	What makes achieving UC difficult	A (too) simple approach
3	Relaxing to ROM	Solution template	Core tool: Succinct Extractable Concrete Commitments
4	Final remarks		

Structure of this talk

1 Quick recap: NIZK | Why UC?

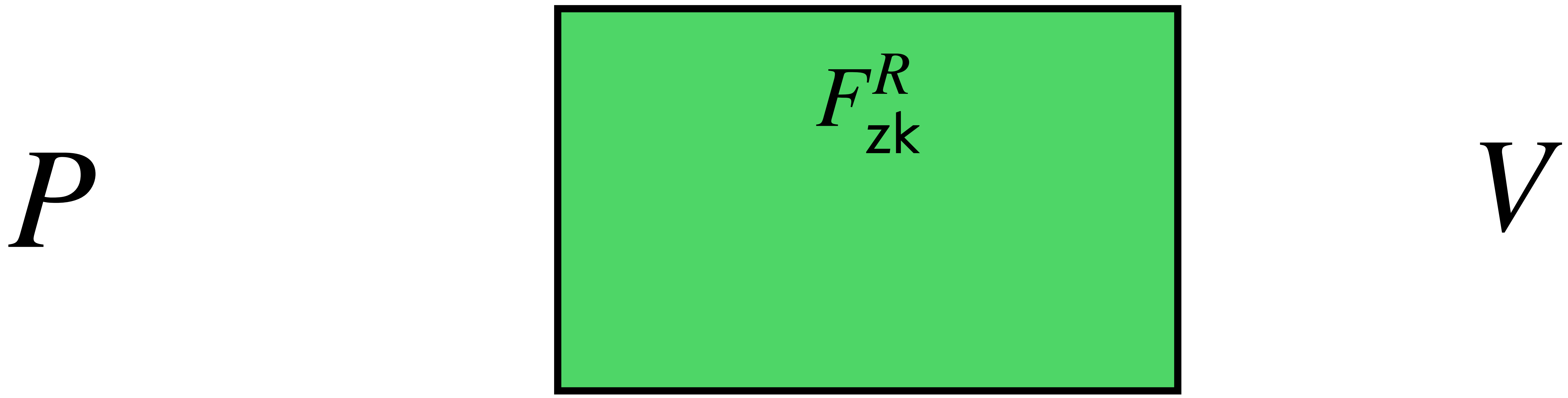
2 What existing works already achieve | What makes achieving UC difficult | A (too) simple approach

3 Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments

4 Final remarks

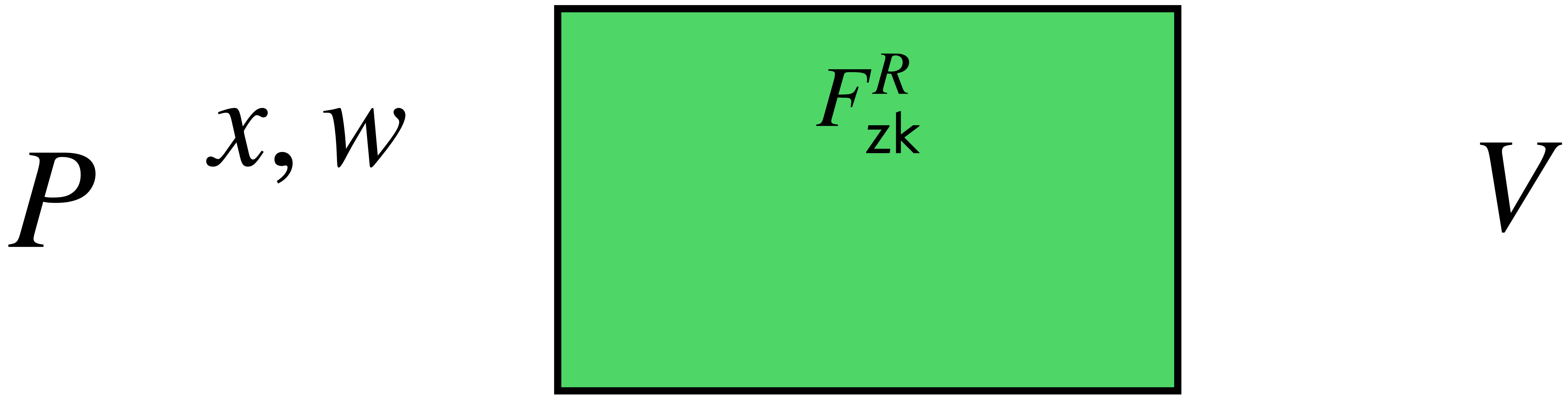
Zero-knowledge in UC

- An ideal oracle that you can use in your higher level protocol
- Safe to compose in any environment



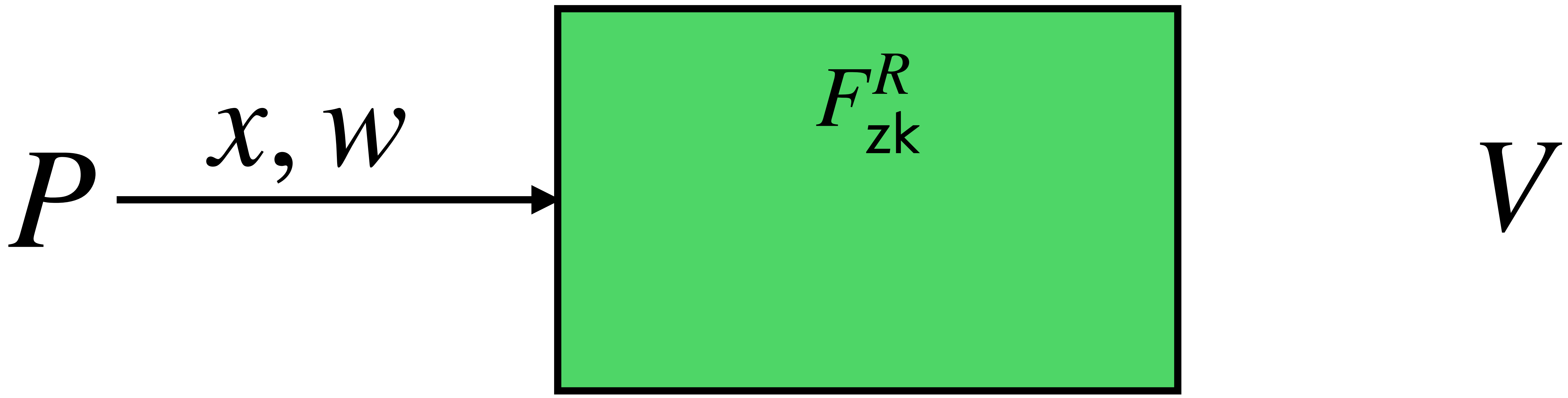
Zero-knowledge in UC

- An ideal oracle that you can use in your higher level protocol
- Safe to compose in any environment



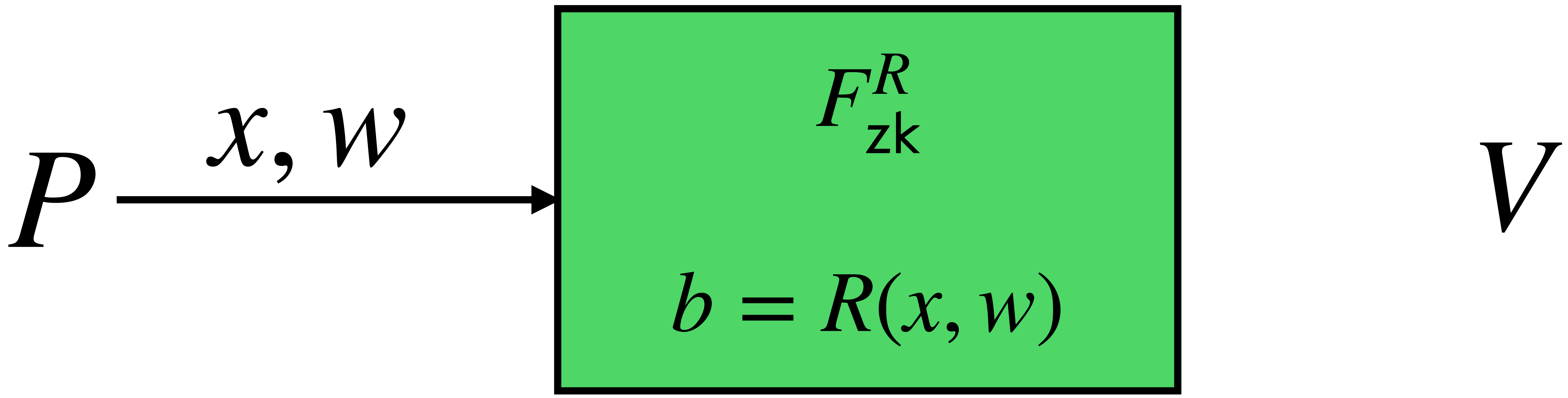
Zero-knowledge in UC

- An ideal oracle that you can use in your higher level protocol
- Safe to compose in any environment



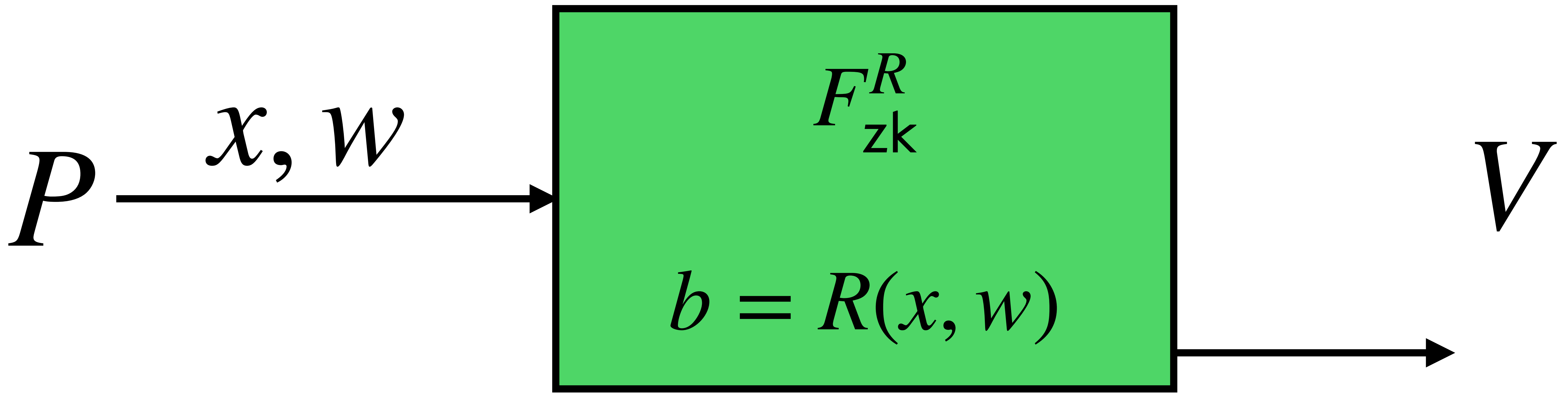
Zero-knowledge in UC

- An ideal oracle that you can use in your higher level protocol
- Safe to compose in any environment



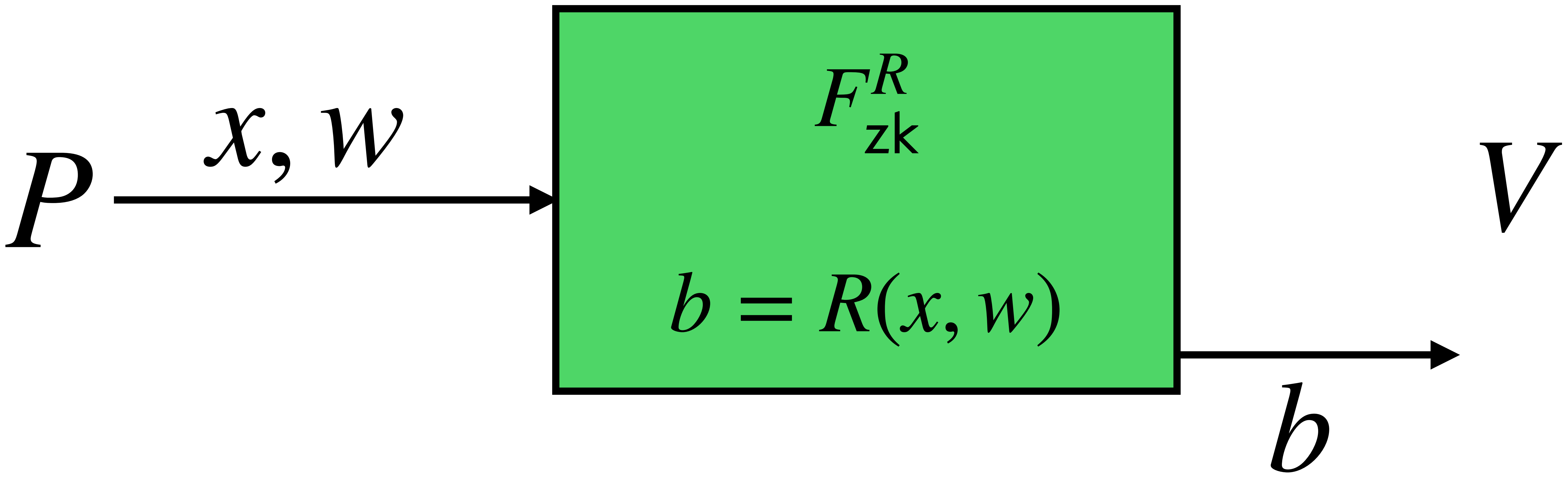
Zero-knowledge in UC

- An ideal oracle that you can use in your higher level protocol
- Safe to compose in any environment



Zero-knowledge in UC

- An ideal oracle that you can use in your higher level protocol
- Safe to compose in any environment



What's Needed for UC Security?

- In a nutshell, simulation and extraction must be **blackbox** and **straight-line**
  - "Knowledge" of a witness may come from a larger protocol context / environment; rewinding the environment or looking at its code is not conducive to proving composition
- Relevant to this talk: Sim and Ext that are straight-line and make oracle use of the adversary

Structure of this talk

1 Quick recap: NIZK | Why UC?

2 What existing works already achieve | What makes achieving UC difficult | A (too) simple approach

3 Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments

4 Final remarks

Structure of this talk

1	Quick recap: NIZK	Why UC?	
2	What existing works already achieve	What makes achieving UC difficult	A (too) simple approach
3	Relaxing to ROM	Solution template	Core tool: Succinct Extractable Concrete Commitments
4	Final remarks		

Starting Point: SE-SNARK

- The strongest non-malleability notion known to be satisfied by SNARKs so far is Simulation Extractability (SE)
- This work is about "lifting" to full UC security
- The difference between SE and UC is subtle; lies in **blackbox extraction**
- In particular, SE-SNARK extractor depends on the code of the adversary—for each adversary $\mathscr{A}$, there exists an extractor $\mathsf{Ext}_{\mathscr{A}}$

Simulation Extractability

$\mathcal{A}$

$\mathsf{Sim}(\mathsf{td}, \cdot\,)$

Simulation Extractability

$\mathscr{A}$

x_i

$\mathsf{Sim}(\mathsf{td}, \cdot\,)$

Simulation Extractability

$$\mathscr{A} \xrightarrow{\quad x_i \quad} \boxed{\mathsf{Sim}(\mathsf{td}, \cdot\,)}$$

Simulation Extractability

$$\mathscr{A} \quad \begin{array}{c} \xrightarrow{\qquad x_i \qquad} \\ \xleftarrow{\qquad\qquad} \end{array} \quad \boxed{\mathrm{Sim}(\mathrm{td}, \cdot\,)}$$

Simulation Extractability

$$\mathscr{A} \quad \underset{\tilde{\pi}_i}{\overset{x_i}{\rightleftarrows}} \quad \boxed{\mathsf{Sim}(\mathsf{td}, \cdot\,)}$$

Simulation Extractability

$\mathscr{A}$ $\xrightarrow{x_i}$ Sim(td, $\cdot$)

$\xleftarrow[\tilde{\pi}_i]{}$

Output $\hat{x}, \hat{\pi}$

Simulation Extractability

$\mathscr{A}$ $\underset{\tilde{\pi}_i}{\overset{x_i}{\rightleftarrows}}$ Sim(td, $\cdot$)

Output $\hat{x}, \hat{\pi}$

- $\mathscr{A}$ wins if $V(\hat{x}, \hat{\pi}) = 1$ but $\mathsf{Ext}_{\mathscr{A}}(\mathsf{td}, \pi)$ fails to output a witness

Non-blackbox Extraction

- SE-SNARK constructions are proven secure with non-falsifiable "knowledge assumptions"

- Roughly, a knowledge assumption purports the existence of an extractor, which can inspect the code of an adversary to deduce useful information

- Eg. Knowledge of Exponent (KEA):
  For any $\mathscr{A}$ s.t. $(X, Y) \leftarrow \mathscr{A}(g, g^a)$ where $X = Y^a$, there exists $\mathsf{Ext}_{\mathscr{A}}$ s.t.
  $y \leftarrow \mathsf{Ext}_{\mathscr{A}}(g, g^a)$ where $g^y = Y$

Structure of this talk

1

Quick recap: NIZK

Why UC?

2

What existing works already achieve

What makes achieving UC difficult

A (too) simple approach

3

Relaxing to ROM

Solution template

Core tool: Succinct Extractable Concrete Commitments

4

Final remarks

Structure of this talk

1 | Quick recap: NIZK | Why UC?

2 | What existing works already achieve | What makes achieving UC difficult | A (too) simple approach

3 | Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments

4 | Final remarks

Non-blackbox Extraction to UC

- The existence of $\mathsf{Ext}_{\mathcal{Z}}$ means that an environment $\mathcal{Z}$ that produces a SNARK must fundamentally know a witness

- However $\mathsf{Ext}_{\mathcal{Z}}$ can not be invoked

- Lifting this SNARK to a UC NIZK is then a matter of forcing the environment to use this knowledge within the protocol context

Simple Approach: Encrypt-then-prove

$P(x, w)$

τ

$V(x)$

$\xrightarrow{\pi}$

Simple Approach: Encrypt-then-prove

$P(x, w)$ $\quad$ τ , pk $\quad$ $V(x)$

$$\xrightarrow{\pi}$$

Simple Approach: Encrypt-then-prove

$P(x, w)$ τ , pk $V(x)$

$$\xrightarrow[\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(w)]{\pi}$$

Simple Approach: Encrypt-then-prove

$P(x, w)$ τ , pk $V(x)$

π : "ct encrypts a witness to x"

$\longrightarrow$

$\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(w)$

Simple Approach: Encrypt-then-prove

$P(x, w)$

τ , pk　　　　Ext(sk, x)

π : "ct encrypts a witness to x"

$\longrightarrow$

$\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(w)$

Simple Approach: Encrypt-then-prove

$P(x, w)$ $\qquad$ τ , $\boxed{\mathsf{pk}}$ $\qquad$ $\mathsf{Ext}(\boxed{\mathsf{sk}}, x)$

π : "ct encrypts a witness to x"

$\longrightarrow$

$\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(w)$

Simple Approach: Encrypt-then-prove

$P(x, w)$

τ , $\boxed{\mathsf{pk}}$ $\quad\quad$ $\mathsf{Ext}(\boxed{\mathsf{sk}}, x)$

π : "ct encrypts a witness to x"

$\longrightarrow$

$\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(w)$

$w = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$

Simple Approach: Encrypt-then-prove

$P(x, w)$

τ , $\boxed{\mathsf{pk}}$ Ext($\boxed{\mathsf{sk}}$, x)

π : "ct encrypts a witness to x"

$\longrightarrow$

$\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(w)$

$w = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$

- Validity of w follows from correctness of encryption+SNARK soundness

Simple Approach: Encrypt-then-prove

$P(x, w)$

τ , $\boxed{\mathsf{pk}}$ $\quad$ $\mathsf{Ext}(\boxed{\mathsf{sk}}, x)$

π : "ct encrypts a witness to x"

$O_\kappa(1)$

$\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(w)$

Circuit succinct $\quad$ $O(|w|)$

$w = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$

- Validity of w follows from correctness of encryption+SNARK soundness

Simple Approach: Encrypt-then-prove

- Approach taken by [DDOPS01] for simulation-sound NIZK, and later CØCØ [KZMQCPRsS15] to obtain circuit-succinct UC SNARK

- However encrypting the witness inherently limits this approach to $\theta(|w|)$ sized proofs

- [KZMQCPRsS15]: “no known UC-secure zero-knowledge proof construction that is circuit *and* witness-succinct, even under non-standard assumptions”

What Assumptions are Reasonable?

- The extractor clearly needs a trapdoor unavailable to the real verifier
- A Common Reference String trapdoor alone is insufficient [CGKS22]

$P(x, w)$

Trusted setup τ $\quad$ $\mathsf{Ext}(\mathsf{td}, x)$

$$\xrightarrow{\pi}$$

What Assumptions are Reasonable?

- The extractor clearly needs a trapdoor unavailable to the real verifier
- A Common Reference String trapdoor alone is insufficient [CGKS22]

$P(x, w)$

Trusted setup $\boxed{\tau}$ $\quad$ $\mathsf{Ext}(\boxed{\mathsf{td}}, x)$

$\xrightarrow{\pi}$

What Assumptions are Reasonable?

- The extractor clearly needs a trapdoor unavailable to the real verifier
- A Common Reference String trapdoor alone is insufficient [CGKS22]

$P(x, w)$

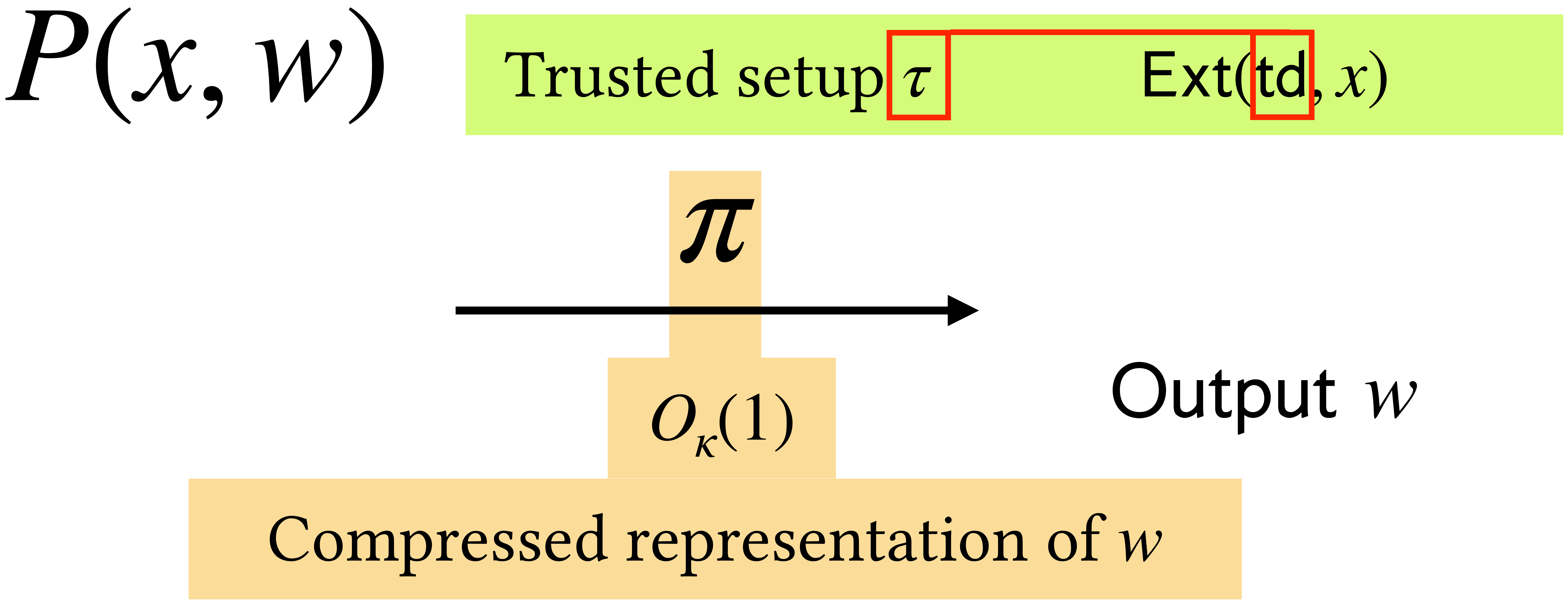
Trusted setup $\boxed{\tau}$ Ext($\boxed{\text{td}}$, x)

$\xrightarrow{\pi}$

Output w

What Assumptions are Reasonable?

- The extractor clearly needs a trapdoor unavailable to the real verifier
- A Common Reference String trapdoor alone is insufficient [CGKS22]



What Assumptions are Reasonable?

- The extractor clearly needs a trapdoor unavailable to the real verifier
- A Common Reference String trapdoor alone is insufficient [CGKS22]
- We need to relax the problem, i.e. grant the extractor further powers/trapdoors (that are still permissible in the UC setting)
- Random Oracle Model is a good fit; easy to model in UC, and practitioners have experience with heuristic instantiations

Structure of this talk

1 Quick recap: NIZK | Why UC?

2 What existing works already achieve | What makes achieving UC difficult | A (too) simple approach

3 Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments

4 Final remarks

Structure of this talk

1 Quick recap: NIZK | Why UC?

2 What existing works already achieve | What makes achieving UC difficult | A (too) simple approach

3 Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments

4 Final remarks

NIZK in the ROM

- P and V additionally make use of a common random oracle H

$$P^H(x, w) \qquad \boxed{\tau} \qquad V^H(x)$$

$$\xrightarrow{\quad\pi\quad}$$

NIZK in the ROM

- P and V additionally make use of a common random oracle H

$$P^H(x, w)$$

$\boxed{\tau}$ $\mathsf{Ext}(\boxed{\mathsf{td}}, \vec{Q}, x)$

$$\xrightarrow{\pi}$$

NIZK in the ROM

- P and V additionally make use of a common random oracle H

$P^H(x, w)$

$\boxed{\tau}$ $\mathsf{Ext}(\boxed{\mathsf{td}}, \vec{Q}, x)$

$\xrightarrow{\pi}$

Output w

NIZK in the ROM

- P and V additionally make use of a common random oracle H

$P^{\boxed{H}}(x, w)$

$\boxed{\tau}$ $\mathsf{Ext}(\boxed{\mathsf{td}}, \boxed{\vec{Q}}, x)$

$\xrightarrow{\pi}$

Output w

NIZK in the ROM

- P and V additionally make use of a common random oracle H

$\vec{Q}$ is the list of queries P made to H

$P^{\boxed{H}}(x, w)$

$\boxed{\tau}$ $\quad$ $\mathsf{Ext}(\boxed{\mathsf{td}}, \boxed{\vec{Q}}, x)$

$\xrightarrow{\pi}$

Output w

Improving the Simple Approach

$P^H(x, w)$

τ , pk

$V^H(x)$

π : "ct encrypts a witness to x"

$O_\kappa(1)$

$\xrightarrow{\hspace{3cm}}$

$\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}}(w)$

$O(|w|)$

Improving the Simple Approach

$P^H(x, w)$ τ , pk $V^H(x)$

π : "ct commits to a witness to x"

$O_\kappa(1)$

$\xrightarrow{\hspace{3cm}}$

$\mathsf{ct} = \mathsf{Commit}(w)$

$O_\kappa(1)$

Improving the Simple Approach

$P^H(x, w)$

τ , pk $\qquad$ $\mathsf{Ext}(\mathsf{td}, \vec{Q}, x)$

π : "ct commits to a witness to x"

$O_\kappa(1)$

$\mathsf{ct} = \mathsf{Commit}(w)$

$O_\kappa(1)$

Now how to extract?

Extracting from the Commitment

- Extractable commitments are straightforward in the ROM:
  - $\mathsf{ct} = H(m, r)$ to commit to m with randomness r
  - Given ct, $\overrightarrow{Q}$: search for $(m, r) \in \overrightarrow{Q}$ such that $H(m, r) = \mathsf{ct}$
- But now "ct commits to a witness to x" is not a well-formed NP statement, as H does not have a circuit description
- <u>Challenge</u>: construct a commitment scheme that is succinct, extractable, and has a meaningful circuit representation

Structure of this talk

1 Quick recap: NIZK | Why UC?

2 What existing works already achieve | What makes achieving UC difficult | A (too) simple approach

3 Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments

4 Final remarks

Structure of this talk

1 — Quick recap: NIZK | Why UC?

2 — What existing works already achieve | What makes achieving UC difficult | A (too) simple approach

3 — Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments

4 — Final remarks

Succinct Extractable Concrete Commitments

- Structure of our commitment: $\mathsf{ct} = (C, \pi_C)$
  - C is a string output by a standard model commitment algorithm Com
  - π_C is a straight-line extractable proof of knowledge of opening of C. i.e. algorithm $\mathsf{Com\text{-}Ext}(\mathsf{td}, \overrightarrow{Q}, \mathsf{ct})$ outputs (m, r), where $\mathsf{Com}(m; r) = C$ when π_C is valid
- Ticks both boxes: "C commits to a witness to x via Com" is a well-formed NP statement, and $\mathsf{Com\text{-}Ext}(\mathsf{td}, \overrightarrow{Q}, \mathsf{ct})$ produces such a witness

Constructing $\mathsf{ct} = (C, \pi_C)$

- P encodes w as the coefficients of a polynomial $f_w \in \mathbb{F}_q[X]$, where $q \in \omega(\mathsf{poly}(\kappa))$ is a parameter of the scheme, and the degree of f_w is determined by the instance

P

Constructing $\mathsf{ct} = (C, \pi_C)$

- P encodes w as the coefficients of a polynomial $f_w \in \mathbb{F}_q[X]$, where $q \in \omega(\mathsf{poly}(\kappa))$ is a parameter of the scheme, and the degree of f_w is determined by the instance

P	$f_w(1)$	$f_w(2)$				$f_w(i)$					$f_w(n)$

Constructing $\mathsf{ct} = (C, \pi_C)$

- P encodes w as the coefficients of a polynomial $f_w \in \mathbb{F}_q[X]$, where $q \in \omega(\mathsf{poly}(\kappa))$ is a parameter of the scheme, and the degree of f_w is determined by the instance

P

p_1	p_2				p_i					p_n
$f_w(1)$	$f_w(2)$				$f_w(i)$					$f_w(n)$

Constructing $\mathsf{ct} = (C, \pi_C)$

- P encodes w as the coefficients of a polynomial $f_w \in \mathbb{F}_q[X]$, where $q \in \omega(\mathsf{poly}(\kappa))$ is a parameter of the scheme, and the degree of f_w is determined by the instance

p_i is a $O_\kappa(1)$ sized proof that $f_w(i)$ is consistent with C

P

p_1	p_2				p_i					p_n
$f_w(1)$	$f_w(2)$				$f_w(i)$					$f_w(n)$

Constructing $\mathsf{ct} = (C, \pi_C)$

- P encodes w as the coefficients of a polynomial $f_w \in \mathbb{F}_q[X]$, where $q \in \omega(\mathsf{poly}(\kappa))$ is a parameter of the scheme, and the degree of f_w is determined by the instance

p_i is a $O_\kappa(1)$ sized proof that $f_w(i)$ is consistent with C

P

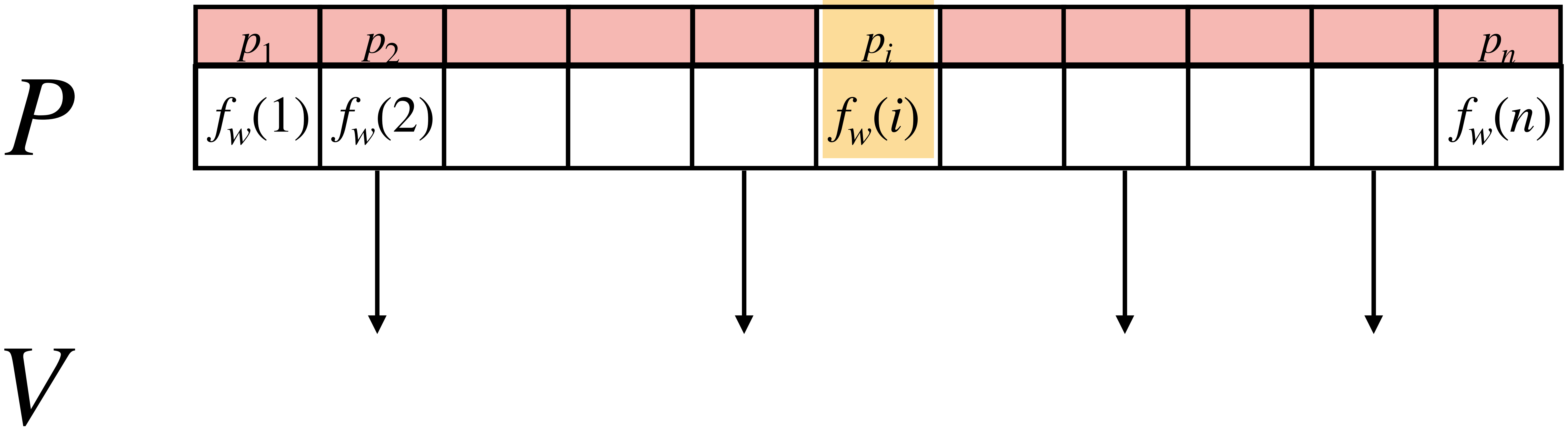
p_1	p_2				p_i					p_n
$f_w(1)$	$f_w(2)$				$f_w(i)$					$f_w(n)$

V

Constructing $\mathsf{ct} = (C, \pi_C)$

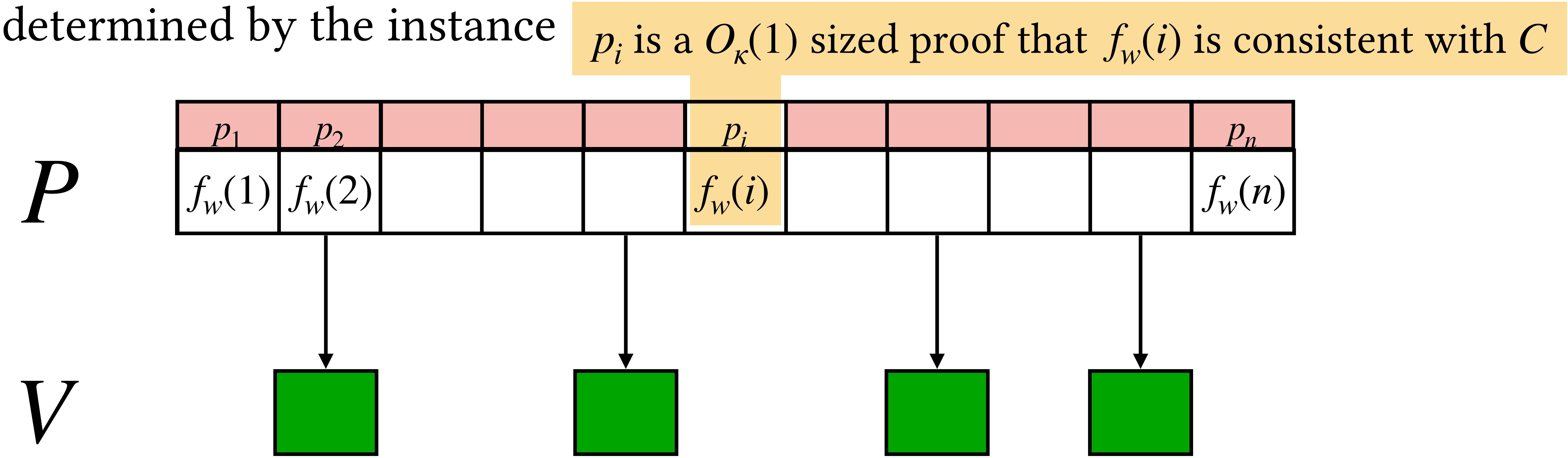
- P encodes w as the coefficients of a polynomial $f_w \in \mathbb{F}_q[X]$, where $q \in \omega(\mathsf{poly}(\kappa))$ is a parameter of the scheme, and the degree of f_w is determined by the instance

p_i is a $O_\kappa(1)$ sized proof that $f_w(i)$ is consistent with C



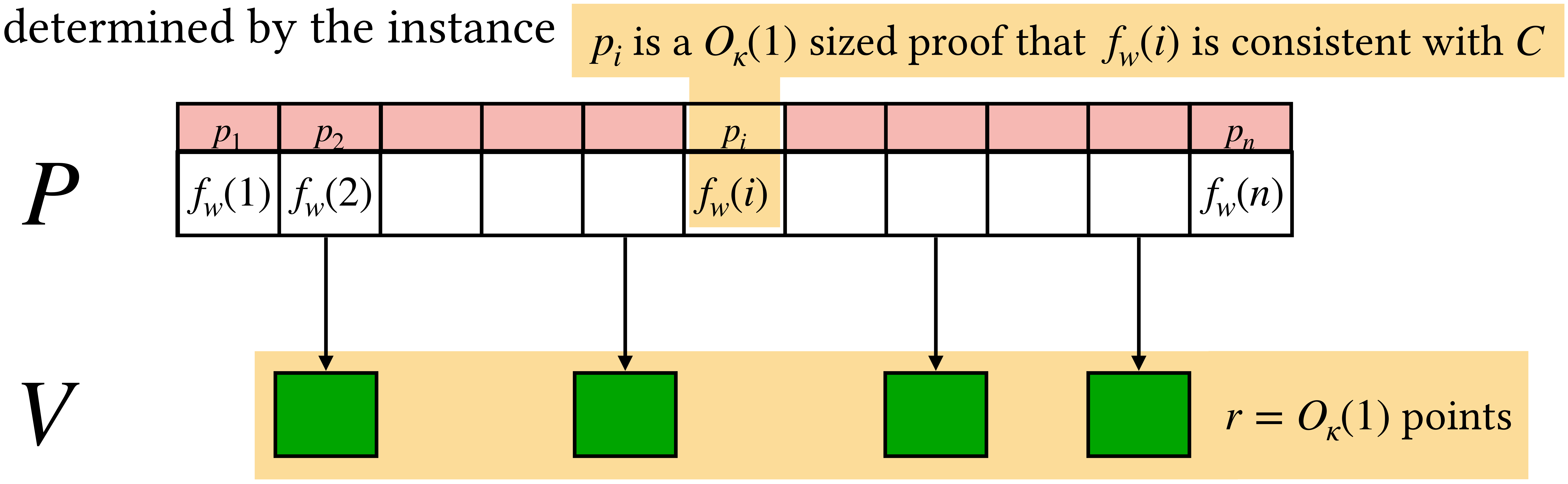
Constructing $\mathsf{ct} = (C, \pi_C)$

- P encodes w as the coefficients of a polynomial $f_w \in \mathbb{F}_q[X]$, where $q \in \omega(\mathsf{poly}(\kappa))$ is a parameter of the scheme, and the degree of f_w is determined by the instance



Constructing $\mathsf{ct} = (C, \pi_C)$

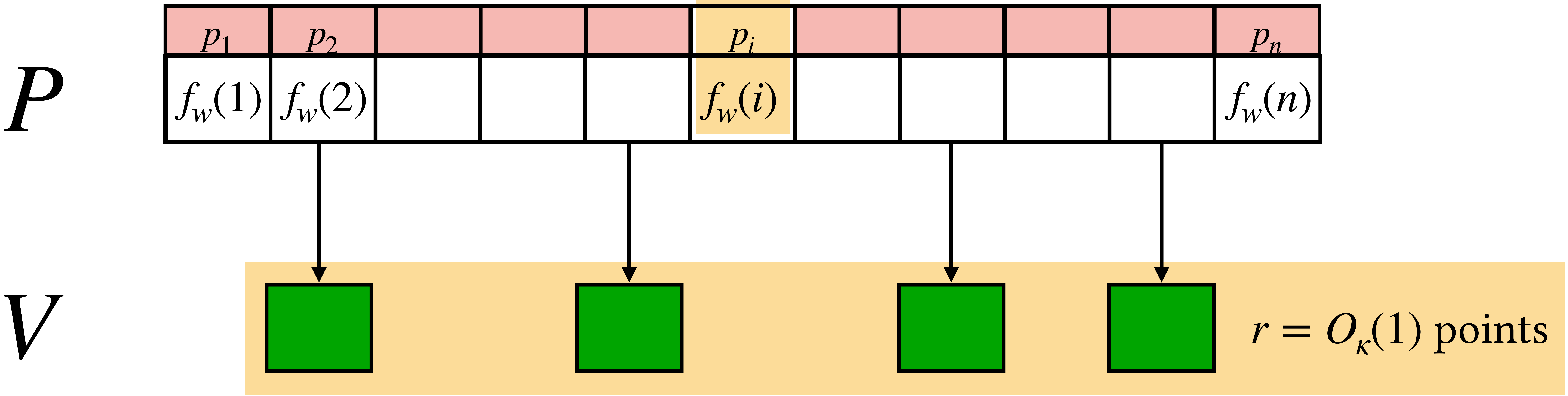
- P encodes w as the coefficients of a polynomial $f_w \in \mathbb{F}_q[X]$, where $q \in \omega(\mathsf{poly}(\kappa))$ is a parameter of the scheme, and the degree of f_w is determined by the instance



Constructing $\mathsf{ct} = (C, \pi_C)$

- P encodes w as the coefficients of a polynomial $f_w \in \mathbb{F}_q[X]$, where $q \in \omega(\mathsf{poly}(\kappa))$ is a parameter of the scheme, and the degree of f_w is determined by the instance

p_i is a $O_\kappa(1)$ sized proof that $f_w(i)$ is consistent with C



Eg. When $r = \kappa$ and $n > 2d$, except with $\Pr < 2^\kappa$ there are at least d correct evaluations of f_w

Compression via [Fischlin 05]

- [Fischlin 05] gives a method for compiling interactive 3 round protocols to straight-line extractable proofs in the ROM

- Achieves more interesting compression properties than simple cut-and-choose, which turns out to be very useful in this setting

P^H $\qquad$ V^H

p_i

$f_w(i)$

$\longrightarrow$

Validate $(C, f_w(i), p_i)$

AND

$H(f_w(i), p_i) \stackrel{?}{=} 0$

Compression via [Fischlin 05]

P

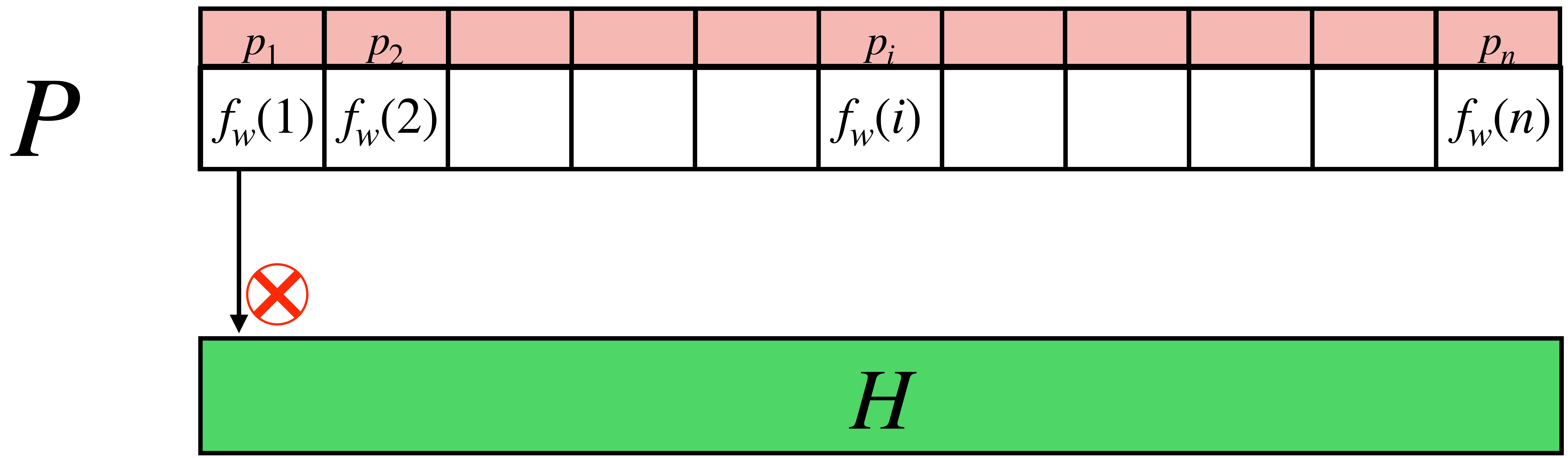
p_1	p_2				p_i					p_n
$f_w(1)$	$f_w(2)$				$f_w(i)$					$f_w(n)$

Compression via [Fischlin 05]

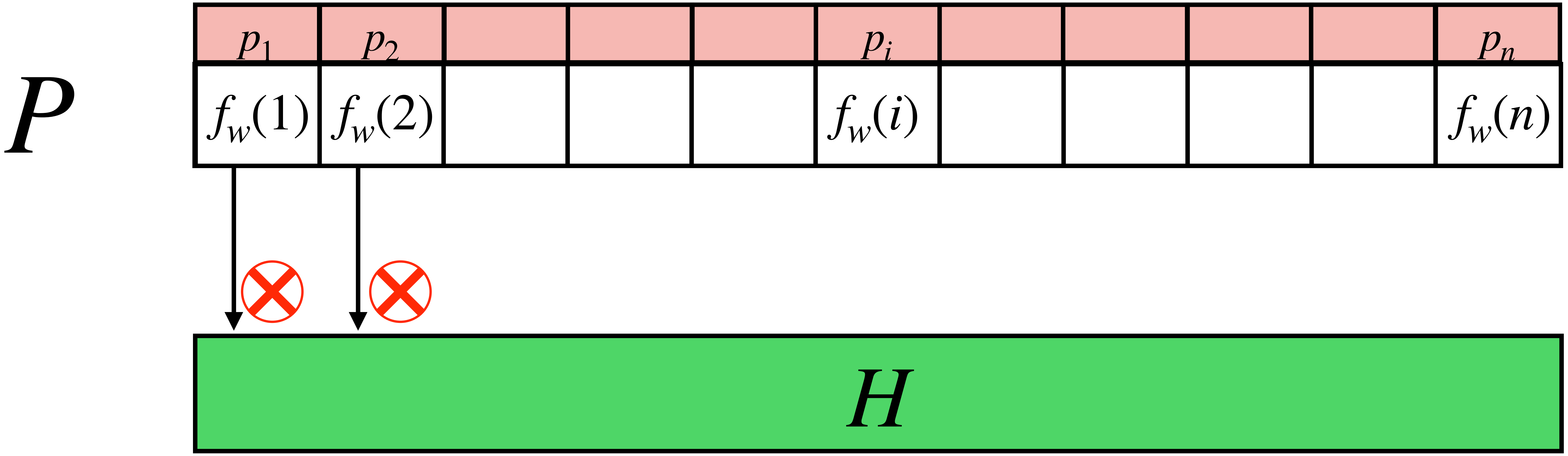
	p_1	p_2				p_i					p_n
P	$f_w(1)$	$f_w(2)$				$f_w(i)$					$f_w(n)$

H

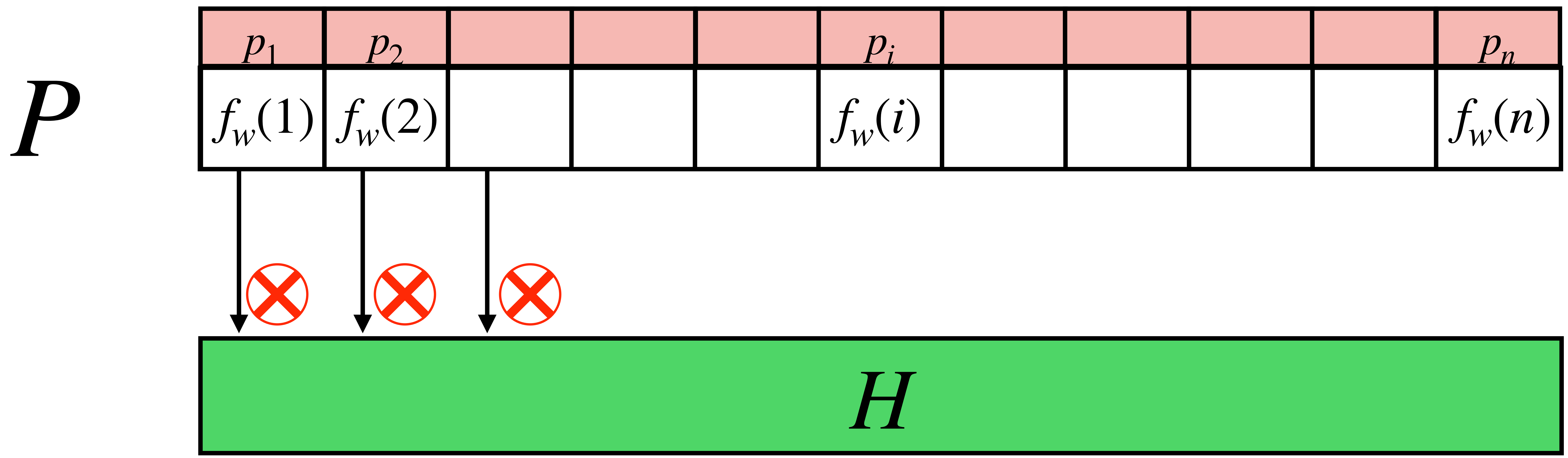
Compression via [Fischlin 05]



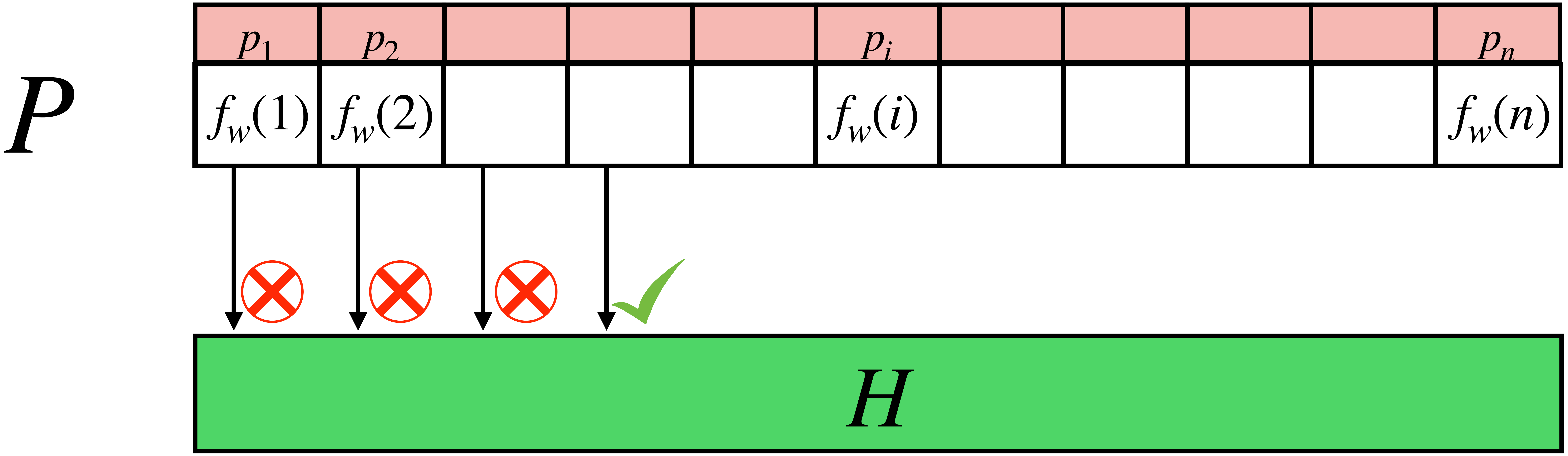
Compression via [Fischlin 05]



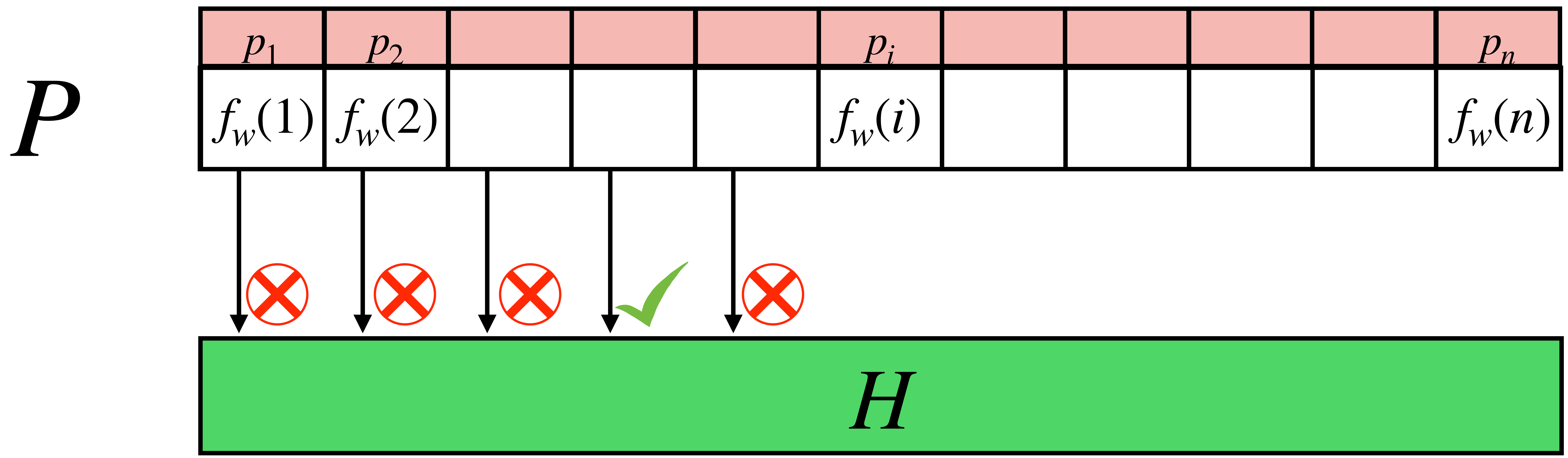
Compression via [Fischlin 05]



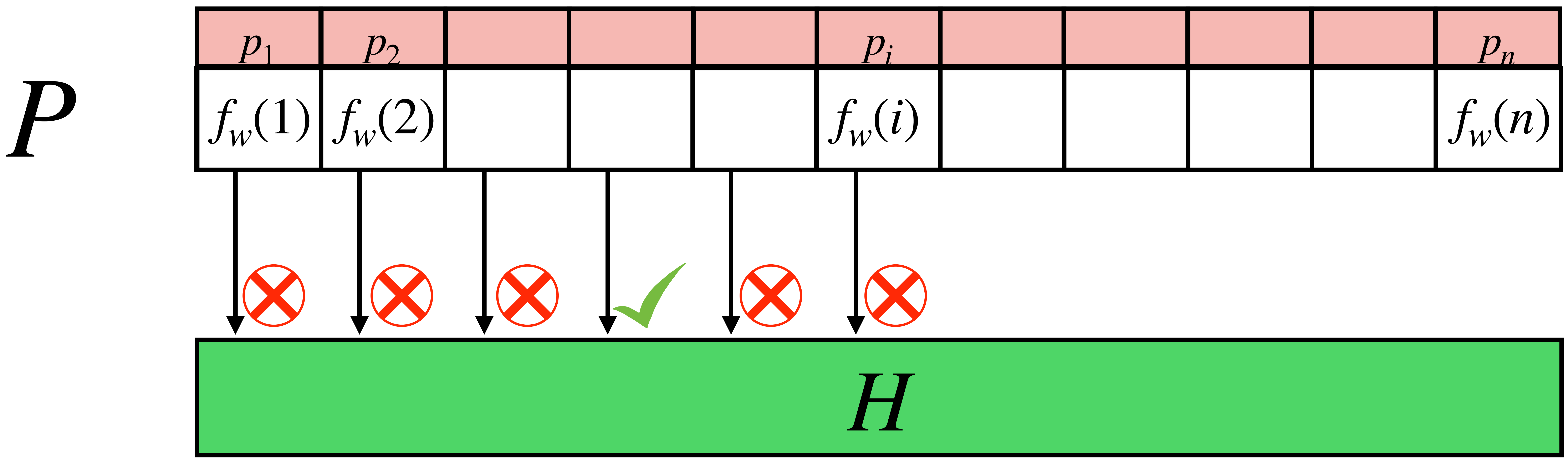
Compression via [Fischlin 05]



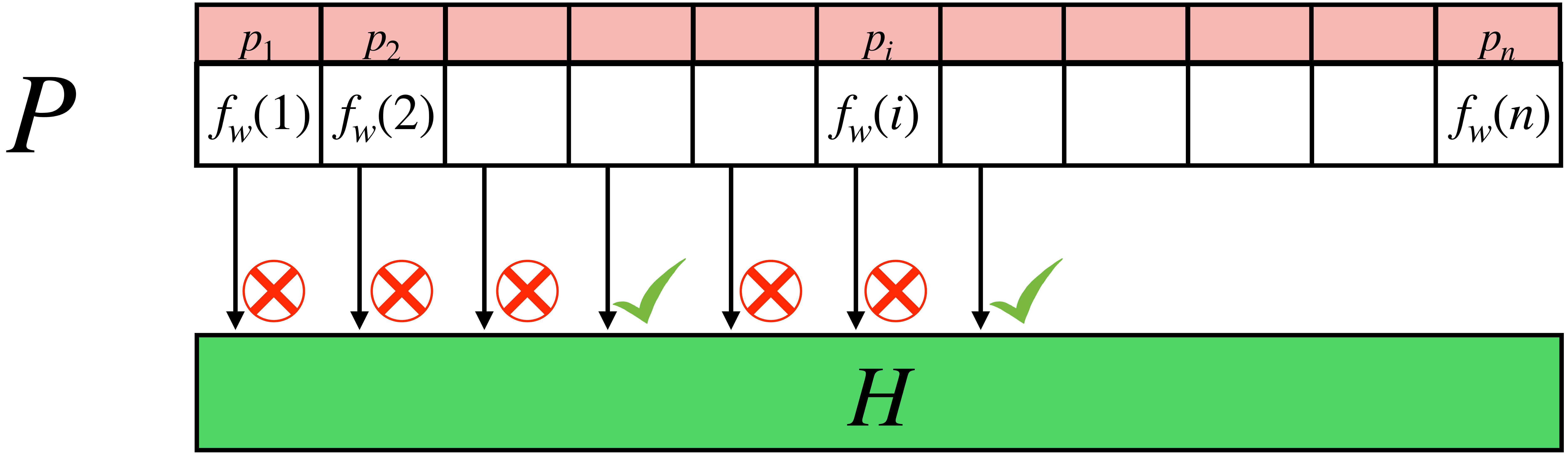
Compression via [Fischlin 05]



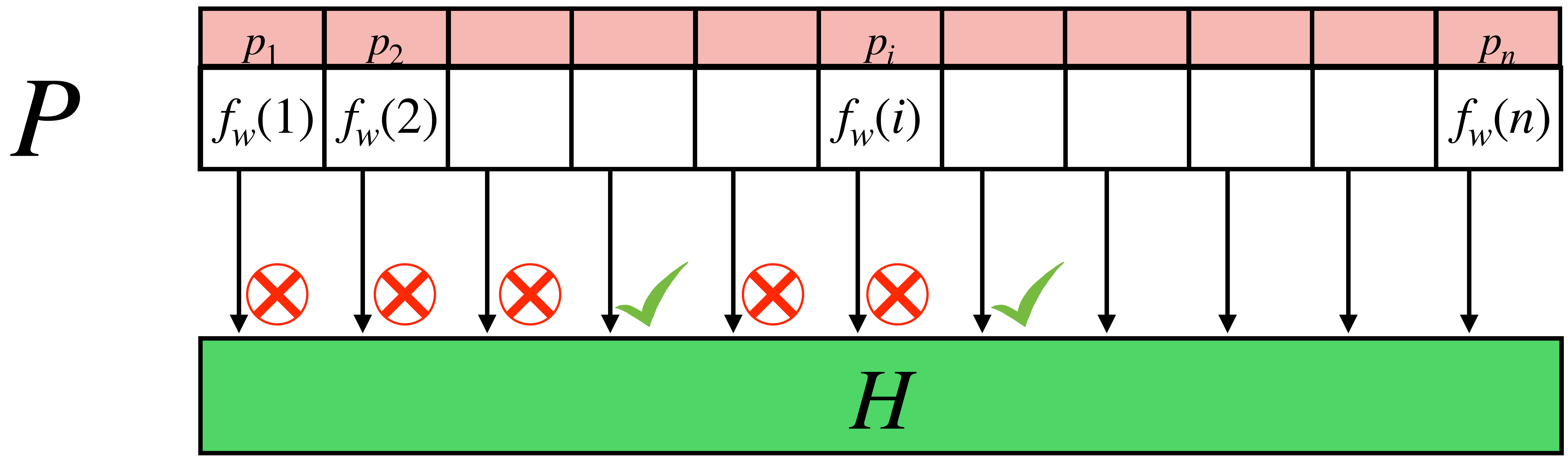
Compression via [Fischlin 05]



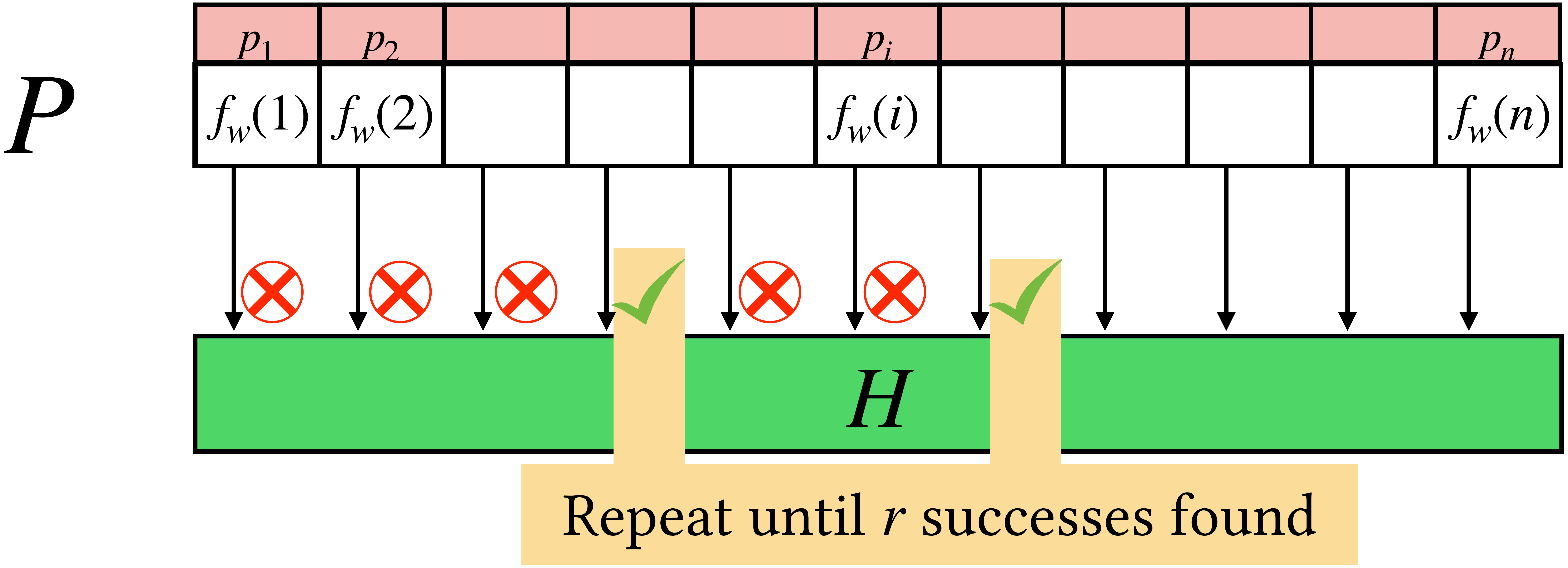
Compression via [Fischlin 05]



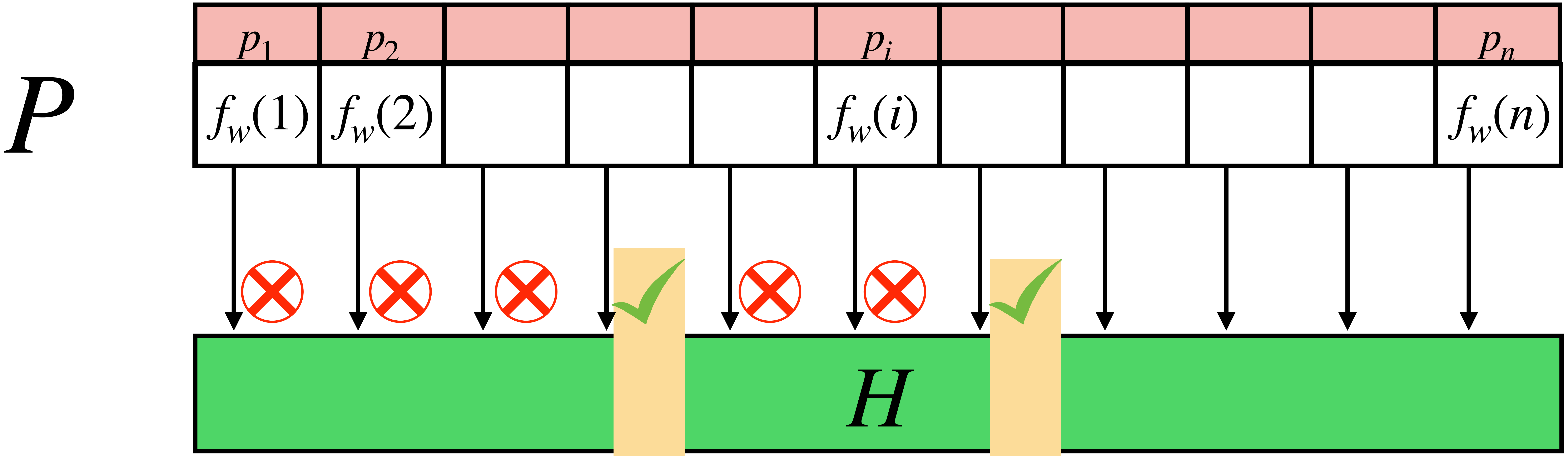
Compression via [Fischlin 05]



Compression via [Fischlin 05]



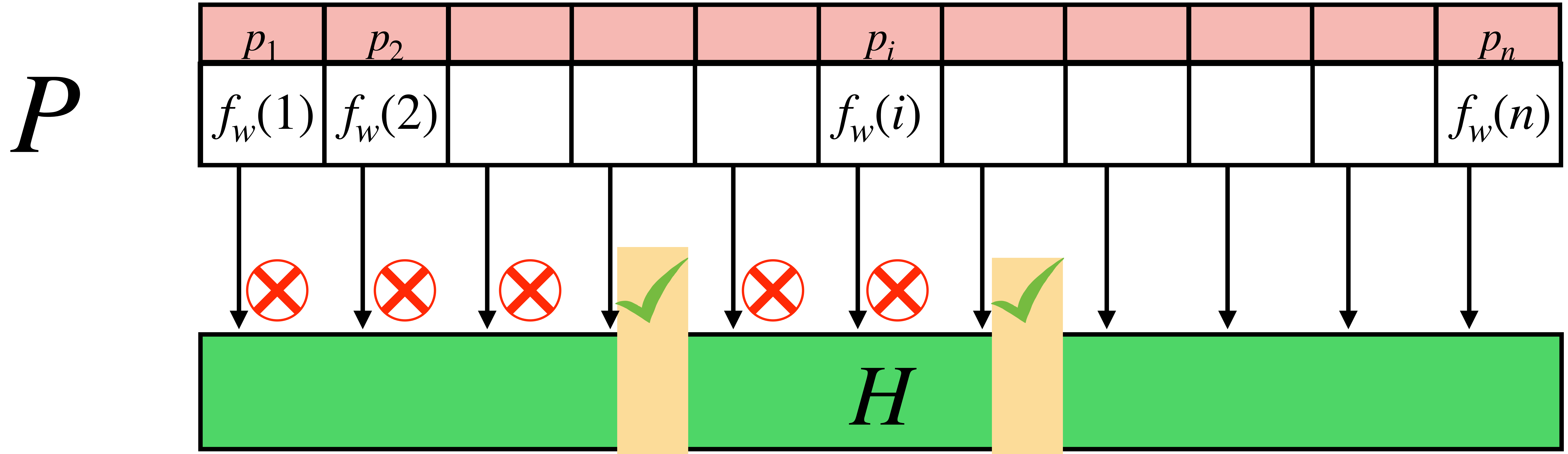
Compression via [Fischlin 05]



Output length of H:
$\log \kappa + \log d$ bits

Repeat until r successes found

Compression via [Fischlin 05]

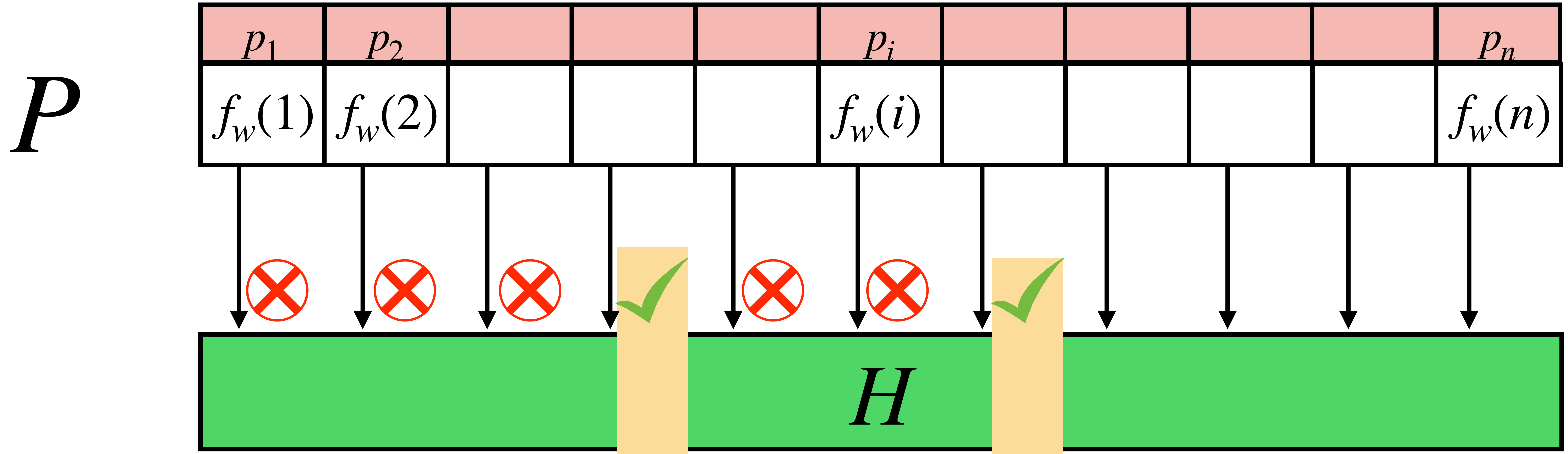


Output length of H: $\log \kappa + \log d$ bits

Repeat until r successes found

Extraction: Except with $\Pr < 2^{-\kappa}$, P is forced to query more than d valid points on f_w to H

Compression via [Fischlin 05]



Output length of H: $\log \kappa + \log d$ bits

Repeat until r successes found

Extraction: Except with $\Pr < 2^{-\kappa}$, P is forced to query more than d valid points on f_w to H

Succinctness: P outputs $r \in O_\kappa(1)$ tuples $(p_i, f_w(i))$

Putting it Together

$P^H(x, w)$

τ $\quad$ $\mathsf{Ext}(\mathsf{td}, \overrightarrow{Q}, x)$

$$\pi : \text{“}C = \mathsf{Com}(w), \text{ and } R(x, w) = 1\text{”}$$

$$\xrightarrow{\hspace{4cm}}$$

$$\mathsf{ct} = (C, \pi_C)$$

Output $\mathsf{Com}\text{-}\mathsf{Ext}(\mathsf{td}, \overrightarrow{Q}, \mathsf{ct})$

Putting it Together

$P^H(x, w)$

τ $\qquad$ $\mathsf{Ext}(\mathsf{td}, \overrightarrow{Q}, x)$

$\boldsymbol{\pi}$: "$C = \mathsf{Com}(w)$, and $R(x, w) = 1$"

$O_\kappa(1)$ $\longrightarrow$

$\mathsf{ct} = (C, \pi_C)$

Output $\mathsf{Com\text{-}Ext}(\mathsf{td}, \overrightarrow{Q}, \mathsf{ct})$

Putting it Together

$P^H(x, w)$

τ $\qquad$ $\mathsf{Ext}(\mathsf{td}, \overrightarrow{Q}, x)$

$\boldsymbol{\pi}$: "$C = \mathsf{Com}(w)$, and $R(x, w) = 1$"

$O_\kappa(1)$

$\longrightarrow$

$\mathsf{ct} = (C, \pi_C)$

$O_\kappa(1)$

Output $\mathsf{Com\text{-}Ext}(\mathsf{td}, \overrightarrow{Q}, \mathsf{ct})$

Putting it Together

$P^H(x, w)$

τ $\mathsf{Ext}(\mathsf{td}, \overrightarrow{Q}, x)$

π : "$C = \mathsf{Com}(w)$, and $R(x, w) = 1$"

$O_\kappa(1)$

$\mathsf{ct} = (C, \pi_C)$

$O_\kappa(1)$ $O_\kappa(1)$

Output $\mathsf{Com}\text{-}\mathsf{Ext}(\mathsf{td}, \overrightarrow{Q}, \mathsf{ct})$

Structure of this talk

1

Quick recap: NIZK

Why UC?

2

What existing works already achieve

What makes achieving UC difficult

A (too) simple approach

3

Relaxing to ROM

Solution template

Core tool: Succinct Extractable Concrete Commitments

4

Final remarks

Structure of this talk

1 | Quick recap: NIZK | Why UC?

2 | What existing works already achieve | What makes achieving UC difficult | A (too) simple approach

3 | Relaxing to ROM | Solution template | Core tool: Succinct Extractable Concrete Commitments

4 | Final remarks

Missing Technicalities

- The final theorem still depends on the non-blackbox extractor (and consequently inherits any knowledge assumptions), albeit in a "UC compatible" way; the NB extractor is only invoked to argue indistinguishability of hybrid experiments

- Zero-knowledge/Simulation requires inflating the degree of f_w

- Applying Fischlin's technique is quite subtle; we need some non-standard *uniqueness* properties from the p_i proofs (satisfied by [KZG10])

In Summary

- We give a compiler (in the ROM) to lift any Simulation Extractable NIZK to a UC NIZK, while preserving the same asymptotic level of succinctness (ignoring security parameter terms)

- Plugging in existing $O_\kappa(1)$ sized proofs, we obtain the first $O_\kappa(1)$ sized UC NIZK

- Our core technical ingredient is to construct a succinct extractable commitment scheme in the ROM via Fischlin's compression method

Questions?